(8)

Working Papers In Speech Recognition
- IV -
The HEARSAY II System

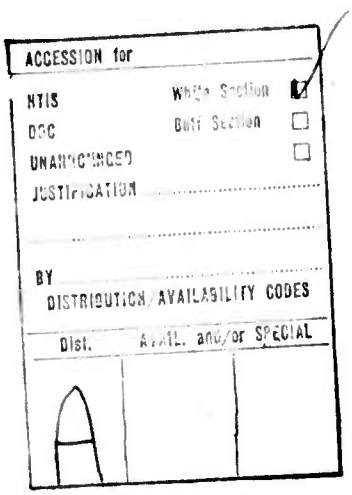
Carnegie-Mellon University
Computer Science Speech Group
February, 1976

DEPARTMENT
of
COMPUTER SCIENCE

Carnegie-Mellon University

MISSING PAGE
NUMBERS ARE BLANK
AND WERE NOT
FILMED

Working Papers In Speech Recognition
- IV -
The HEARSAY II System



Carnegie-Mellon University
Computer Science Speech Group
February, 1976

Pittsburgh, PA 15213

This research was supported by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

PREFACE

This volume contains several papers describing the current status of the Hearsay II system. Some of these papers have appeared elsewhere in the literature and are included here for convenience. The relevant references appear in the table of contents.

Hearsay II has been the major research effort of the CMU speech group over the past three years. During this period, solutions were devised to many difficult conceptual problems that arose during the implementation of Hearsay I and other earlier efforts. The result represents not only an interesting system design for speech understanding but also an experiment in the area of knowledge-based systems architecture. Attempts are being made by other AI groups to use this type of systems architecture in image understanding and other knowledge-intensive systems.

The Hearsay II effort is headed by Lee Erman. The other major participants include Frederick Hayes-Roth, Victor Lesser, and Linda Shockey. The Hearsay II group consists of 7 full-time researchers and four graduate students. Several others in the CMU environment participate in the effort. The group is loosely organized into subgroups:

System design (Lesser, Gill, McKeown, Erman)
Focus of attention and policy (Hayes-Roth, Lesser)
Syntax, Semantics, and Discourse (Hayes-Roth, Mostow, Gill)
Word hypothesization and Verification (Erman, Cronk, Smith)
Phonetics (Shockey, Adam)
Segmentation and Labeling (Goldberg, Reddy)
Performance analysis (Erman, Reddy, Masulis)
Data base (Shockey)

The various papers appearing in this volume are representative of the present state of conceptualization and implementation of the system. The first paper provides an overview of much of this research. The system has been operational for some time and most of the effort at present involves performance analysis, acquisition and representation of additional knowledge, and experimentation with different control strategies.

Raj Reddy

page

TABLE OF CONTENTS

Overview of the Hearsay Speech Understanding Research

Lee D. Erman

Hearsay is the generic name for much of the speech understanding research in the computer science department at Carnegie-Mellon University (CMU). The major goals of this research include the investigation of computer knowledge-based problem-solving systems and the practical implementation of speech input to computers. An emphasis of this effort is the design of system structures for efficient implementation of such systems.

We will first describe the problem of speech understanding and (in Section 2) present the context of the Hearsay effort. Section 3 describes the Hearsay model and implementation philosophy. Then, In Section 4, Hearsayl is described, including some major design limitations which formed much of the motivation for Hearsayll, described in Section 5.

I. The Problem of Speech Understanding

In order to provide a framework for discussion, a conceptual model of speech communication is presented:

1) The purpose of a speech utterance is to transmit information from the speaker to the listener.
2) The speaker starts with some deep semantic representation of the message. Several kinds of transformations are applied to this representation (syntactic, linguistic, phonological, neurological, articulatory, acoustic, etc.). The result is an acoustic signal.
3) The acoustic signal is detected by the listener. The listener applies transformations which are similar (though inverse) to those of the speaker; the result is some semantic representation for the listener.
4) The correctness or effectiveness of the transmission is related to the correspondence between the meaning that the speaker intends and the meaning derived by the listener; it is measured in behavioral terms—i.e., what actions of the listener are triggered by receipt of the message. (Very often this behavior takes the form of an utterance generated by the original listener.)

The goal of automatic speech understanding is to produce a machine (usually in the form of a computer program) which can effectively perform as the listener.

The problem of understanding speech with the competence of a human is formidable. A reasonable plan is to approach the most general kinds of solutions by designing and building a sequence of systems, each of which is more ambitious than the previous. There are many dimensions along which to move to provide this graded sequence (e.g., requirements of vocabulary size, speed of response, accuracy, number of speakers). A way of capturing these various dimensions is the concept of a *task*—a well-defined domain within which the machine is to perform some functions. For example, the task might be to answer the user's (speaker's) questions about airline flight schedules or to provide an interactive computer-programming facility. In defining a task, one important aspect is the spoken *input language*. This language is pre-specified lexically, syntactically, and semantically; that is, descriptions are given of the words, how they may be sequenced to form sentences, and the meaning of the sentences in the context of the task.[1]

There are two major aspects of the speech communication process which generate most of the problems in machine understanding:

1) *The nature of the speech signal*—The transformations involved in speech production are many and complex, and they strongly interact with each other. The result is a very large amount of *variability* in the signal which conveys little or no meaning, i.e., which is noise in the context of the speech understanding task. Repetitions of the "same" utterance, spoken by one speaker under unchanging conditions just seconds apart often

1 This use of a task to constrain the problem is not as artificial as it may first appear. Usually human speech understanding is also performed in "constrained domains"—In almost any given situation only a small subset of all possible messages is likely.

result in significant variation of the signal. As the various conditions (e.g., identity, age, gender, emotional state, and environment of the speaker) are relaxed and allowed to change, this variability increases significantly. Further, strong interactions occur among the various elements; that is, words, phones, phrases, etc., influence and modify nearby words, phones, phrases, etc., and thus have differing manifestations in different contexts.[2]

2) *The nature of our knowledge of the transformations*—Theories which attempt to explain the production of speech are, in general, incomplete and inadequate in explaining the phenomena with a great deal of accuracy. Also, it is often difficult to translate existing theories into the framework of feasible recognition algorithms.

Largely because of these two aspects, the kinds of machine speech understanding systems developed can be characterized as having several interesting and problem-laden features:

1) The system must make use of multiple and diverse *sources of knowledge* to solve the problem (e.g., acoustic-phonetics, phonology, syntax, semantics, pragmatics); these knowledge sources (*KSs*) correspond to the different kinds of transformations that generate the speech signal. Designing an effective control structure for these many diverse KSs is crucial and difficult.

2) Each source of knowledge is *incomplete and errorful*. Thus, although it is used in an attempt to further the recognition of an utterance, each KS will also introduce errors into the analysis process. The different sources must work to correct each other's mistakes in order to keep errors from propagating excessively.

3) The systems developed tend to be *large and complex*. Building, debugging, understanding, and evaluating them is difficult. In particular, many researchers need to interact with the system over a period of several years, both experimenting with its operation and modifying it. An important aspect of system modification is the ability to modify and replace individual KSs.

4) Because of the effectiveness and apparent ease of human performance in the speech understanding task, a useful solution to the problem must be a system which approaches that *performance*, primarily in terms of speed, accuracy, and, ultimately, economy.

5) Because the systems tend to be highly experimental, they must be exercised often and over substantial amounts of trial data. The performance of the system *while under development* (particularly in terms of speed of execution) is an important factor in determining how much experimentation can occur. Thus, issues of performance are crucial even in the development stage.

6) Because the systems are complex and experimental, the interface through which the researcher controls and interacts with the system is crucial. The researcher must be able to interact with the system flexibly and at the functional level of the system (in addition to the more traditional machine language and programming language levels).

This has been a short introduction into the problems of developing speech understanding systems. A more complete analysis of the problem, including pointers to the relevant literature, can be found in Newell et al. [71].

2. Context of This Work

Hearsay's direct lineage can be traced back ten years. The work of Reddy and Reddy & Vicens at Stanford University (Reddy [66]; Reddy and Vicens [68]; Vicens [69]) resulted in extending the state-of-the-art of isolated word recognition systems (e.g., 91% accuracy on a 561-word lexicon in ten times real-time on a PDP10 and with live input). This system differed from most earlier ones, which were essentially pattern classifiers, in that it contained a substantial amount of speech knowledge and it used extensive heuristics in applying the knowledge to prune the search space. In addition, one version of the system was created which used syntactic constraints and operated on connected speech (although in a very *ad hoc* and unextendable manner).

The Hearsay model for speech understanding was developed at CMU during 1970-1971 (Reddy, Erman, and Neely [70]; Reddy [71]; Reddy, Erman, and Neely [72]). This model faced the problems of speech understanding (i.e., in a task domain) and connected speech. The Hearsayl system was designed and built as an implementation of this model (Reddy, Erman, and Neely [73]; Reddy, Erman, Fennell, and Neely [73]; Neely [73]; Erman [74]). This system, which was the first demonstrable live system to handle non-trivial connected speech, became operational in June, 1972, and has been since augmented and studied (Lowerre [75]). Although a number of simplifying assumptions were made in implementing the model, Hearsayl does address the problems of connected speech and of the role and interactions of different kinds of knowledge. By exhibiting a successfully working system which is

2 We are concerned here with *connected speech* input, as opposed to *isolated word* systems in which the words (for short phrases treated as indivisible units) are spoken individually.

based on a model and by providing a set of solutions to these problems (even if some of the solutions are known to be far from optimal), Hearsayl clarified the problems and serves as a basis, and encouragement, for subsequent work.

The experience of building and experimenting with Hearsayl, together with the other research in the field, led to a design review which resulted in the Hearsayll system (Lesser, Fennell, Erman, and Reddy [74]). Hearsayll is also based on the Hearsay model; it generalizes and extends many of the concepts which exist in a more simplified form in the Hearsayl system.

Concurrent with the early stages of the Hearsay development, a group was formed by the Advanced Research Projects Agency (ARPA) to study the feasibility of developing speech understanding systems. This group, which included researchers active in artificial intelligence as well as those in more traditional directions of speech recognition research, produced its report in May, 1971. This report (Newell et al. [71]) provides a comprehensive and detailed analysis of the problems involved. Part of this study included the specification of a set of nineteen dimensions for describing the capabilities of a speech understanding system—the first column of Figure 1 summarizes those dimensions.

On recommendation of the study group, a five-year ARPA Speech Understanding Research effort was launched in October, 1971. An innovative plan with five principal contractors (including CMU) was chosen; each was to aim to produce a complete system meeting a set of specifications laid out by the study group (the second column of Figure 1) and all were to interact, exchanging ideas and data. Although charged to meet the same set of specifications, each group was free to choose its own orientation (and task domains). Thus, the flavor of each of the systems reflects the particular expertises and motivations of the people involved.

The Hearsay research represents CMU's major efforts to meet the ARPA specifications; in particular, it is hoped that Hearsayll will accomplish that goal. In addition, several other systems are being experimented with, also aiming to meet these goals: a version of the Dragon system (Baker [75]) being extended by Reddy and Lowerre and a combination of Hearsayl and Dragon (Lowerre [75]).

In this paper we will describe only the Hearsay effort. An IEEE symposium on speech recognition was held at CMU in April, 1974, at which most workers in the field were represented. The contributed and invited papers from that symposium (Erman [74b]; Reddy [75]) provide a comprehensive description of the state-of-the-art at that time.

3. The Hearsay Model and Implementation Philosophy

This section describes a general model of speech understanding, the "Hearsay model", and some of the problems implied by that model. The following two sections provide overviews of the Hearsayl and Hearsayll implementations of that model.

As one knowledge source (KS) makes errors and creates ambiguities, other KSs must be brought to bear to correct and clarify those actions. This KS cooperation should occur as soon as possible after the introduction of an error or amoiguity in order to limit its ramifications. The mechanism used for providing this high degree of cooperation is the *hypothesize-and-test* paradigm. In this paradigm, solution-finding is viewed as an iterative process. Two kinds of KS actions occur: a) the creation of an *hypothesis*, an "educated guess" about some aspect of the problem, and b) tests of the plausibility of the hypothesis. For both of these steps, the KS uses a *priori* knowledge about the problem, as well as the previously generated hypotheses. This "iterative guess-building" terminates when a consistent subset of hypotheses is generated which satisfies some specified requirements for an overall solution.

As a strategy for developing such systems, one needs the ability to add and replace KSs and to explore different control strategies. Thus, such changes must be relatively easy to accomplish; there must also be ways to evaluate the performance of the system in general and the roles of the various KSs and control strategies in particular. This ability to experiment conveniently with the system is crucial if the amount of knowledge is large and many people are needed to introduce and validate it. One means of helping to provide these flexibilities is to require that KSs be independent; i.e., the explicit interactions between KSs and their assumptions about each other must be minimal.

Besides providing for the modification and evaluation of KSs, decomposition of the system into relatively independent KSs also facilitates its implementation on an asynchronous multi-processor machine. Such configurations seem increasingly attractive as cost-effective ways of obtaining large amounts of computing power. One problem that has limited the development and usage of such machines is the difficulty of decomposing large problems for such machines. Erman, Fennell, Lesser, and Reddy [73] describe this problem and outline some early solutions in the Hearsay context; Lesser [75] provides a survey of this subject.

The basic view of development of a speech understanding system includes a strong component of experimentation: one needs to build a system and

Dimensions and Examples	ARPA Specifications for 1976 Systems *The system should.*
(1) *Manner of Speech* connected? isolated words?	(1) accept connected speech
(2) *Number of Speakers* one? small set? open population?	(2) from many
(3) *Dialect and Manner* cooperative? casual? single gender? both genders? children? what dialect(s)?	(3) cooperative speakers of the "general American dialect",
(4) *Environmental Conditions* quiet room? computer room? factory? public place?	(4) in a quiet room
(5) *Transducer* high quality microphone? telephone?	(5) over a good quality microphone
(6) *Speaker Tuning* few sentences? paragraphs? full vocabulary?	(6) allowing slight tuning of the system per speaker,
(7) *Speaker Training* natural adaptation? elaborate?	(7) but requiring only natural adaptation by the user,
(8) *Vocabulary Size and Selection* 50? 200? 1,000? 10,000? preselected? selective rejection? free?	(8) permitting a slightly selected vocabulary of 1,000 words,
(9) *Grammar* fixed phrases? artificial language? free English? adaptable?	(9) with a highly artificial syntax,
(10) *Task* highly constrained (e.g., simple retrieval)? focussed (e.g., numerical algorithms)? open?	(10) and a task with a constrained and fairly simple semantics,
(11) *User Model* nothing? current knowledge about the user?	(11) with a simple psychological model of the user,
(12) *Mode of Interaction* response only? ask for repetitions? explain language? discuss communications?	(12) providing graceful interaction,
(13) *Error Rate* none (<0.1%)? <10%? >20%?	(13) tolerating less than 10% semantic error,
(14) *Response Time* no hurry? few times real-time? immediate?	(14) in a few times real-time,
(15) *Processing Power* 1×10^6 instructions/sec? 10 mips? 100 mips? 1000 mips?	(15)
(16) *Memory Size* 1 megabit? 10mb? 100mb? 1000mb?	(16)
(17) *System Organization* simple program? multiprocessing? parallel processing? unidirectional processing? feedback? backtrack? planning?	(17)
(18) *Cost* $0.001/sec. of speech? $0.01/s? $0.1/s? $1.0/s?	(18)
(19) *Operational Date*	(19) and have a prototype demonstrable in 1976.

Figure 1: Dimensions of Speech Understanding Systems and ARPA Specifications for 1976. (After Newell et al. [71].)

then manipulate and study it. In order to provide an environment to accomplish this, a two-level approach is taken: First, a basic set of facilities is provided, and, second, various configurations are built using these facilities. These facilities, which together are called the *kernel*, form a problem-dependent programming system for building and experimenting with particular configurations. The correct choice of kernel facilities and their implementation are crucial ingredients in developing a system.

The Blackboard—Representation of Knowledge

The requirement that KSs be independent implies that the functioning (and very existence) of each must not be necessary or crucial to the others. On the other hand, the KSs are required to cooperate in the iterative guess-building, using and correcting one another's guesses; this implies that there must be interaction among the processes. These two opposing requirements have led to a design in which each KS interfaces to the others externally in a uniform way that is identical across KSs and in which no knowledge source knows what or how many other KSs exist. The interface is implemented as a dynamic global data structure, called the *blackboard*.[3] The primary units in the blackboard are the guesses about particular aspects of the problem—the hypotheses. At any time, the blackboard holds the current state of the system; it contains all the guesses about the problem that exist. Subsets of hypotheses represent partial solutions to the entire problem; these may compete with the partial solutions represented by other (perhaps overlapping) subsets.

Each KS may access information in the blackboard. Each may add information to the blackboard by creating (or deleting) hypotheses, by modifying existing hypotheses, and by establishing or modifying explicit structural relationships among hypotheses. The generation and modification of globally accessible hypotheses is the exclusive means of communication among the diverse KSs. This mechanism of cooperation, which is an implementation of the hypothesize-and-test paradigm, allows a KS to contribute knowledge without being aware of which other KSs will use the information or which KS supplied the information that it used. It is in this way that KSs are made independent and separable. The structural relationships form a network of the hypotheses and are used to represent the deductions and inferences which caused a KS to generate one hypothesis from others. The explicit retention in the blackboard of these dependency relationships is used to hold, among other things, competing hypotheses.

Because of the central importance of the blackboard, its design (i.e., the design of the structure of hypotheses and their relationships) is crucial. This is usually called the problem of *representation*.

Activation of Knowledge Sources—Focus of Attention

An action of a KS in the blackboard takes place in the context of some hypotheses already existing in the blackboard. For example, a KS which hypothesizes words may require a stressed vowel (as well as some surrounding sounds) as its context in order to consider generating new word hypotheses.

At any time there may be many different contexts which satisfy the needs of one or more KSs. The problem of choosing the order for activating KSs on contexts is generally called the problem of *control flow*. Because there may be many such possible activations and because each activation of a KS will, in general, create the potential for even more activations (e.g., the word hypothesizer, given a single new stressed vowel context, might hypothesize five new words as competing candidates—each of these might provide a new context for a syntactic parser), the number of possible activations may grow.

If very, very large amounts of processing power (and memory) were available, one could consider actually activating all KSs in all their possible contexts. This would expand the blackboard with many (competing) hypotheses. Assuming this would eventually terminate (i.e., at some point no new contexts are created), a decision process could then try to pick from all the competing hypotheses that subset which best describes the data—this would be the system's "solution" to the problem. Because of this combinatoric explosion of possibilities (caused mostly by the problems of variability and incompleteness in the signal and errorfulness of the KSs), this complete expansion is not feasible. Therefore, the control strategy can pick only a small subset of the applicable KS activations; this can be thought of as exploring a limited portion of the (potential) fully-expanded blackboard. The problem of choosing a control strategy which can efficiently reach the correct set of hypotheses is called the *attention-focusing* problem. Its solution is also critical for the success of a system: If portions of the correct solution are pruned, the solution will never

3 The term "blackboard" was used by Simon [66] in describing a mechanism in long-term memory as part of a theory of the psychology of problem-solving. Simon [71] further develops this concept and elaborates its uses in the context of an abstract model for problem-solving.

be found; if many incorrect portions are not pruned, the combinatoric explosion will use large amounts of computing resources (and may also force the system to give up before reaching the solution).

The problems of representation of knowledge and searching a large solution space (focus of attention) are two of the central problems of artificial intelligence. The speech understanding problem, with its requirements of high performance and the use of diverse and errorful KSs, provides a rich field for their study.

4. Overview of Hearsayl

The blackboard of Hearsayl consists of partial sentence hypotheses, each of which is a sequence of words with non-overlapping time locations in the utterance. Each is a *partial* sentence hypothesis because not all of the utterance need be described by the given sequence of words. In particular, gaps of one or more words of the utterance which have not yet been hypothesized (in the context of the particular sentence hypothesis) are designated by "filler" words. The partial sentence hypotheses also contain confidence ratings for each word hypothesis and a composite rating for the overall sequence of words. A sentence hypothesis is the focal point that is used to invoke a KS. The sentence hypothesis also contains the accumulation of all information that all KSs have contributed to that hypothesis.

System activity goes through a number of cycles. In each cycle there is one partial sentence hypothesis on the blackboard which is the focal point of activity; this focal hypothesis forms the context for KS activity during the cycle. KSs are activated in a lockstep sequence consisting of three phrases per cycle: *poll*, *hypothesize*, and *test*. At each phase, all KSs are activated for that phase, and the next phase does not commence until all KSs have completed the current one. The *poll* phase involves determining which KSs have something to contribute to the focal sentence hypothesis; polling also determines how confident each KS is about its proposed contributions. The *hypothesize* phase consists of activating the KS showing the most confidence about its proposed contribution of information. This KS then hypothesizes a set of possible words (option words) for some (one) "filler" word in the speech utterance. The *testing* phase consists of each KS evaluating (verifying) the possible option words with respect to the given context. After all KSs have completed their verifications, the option words which seem most likely, based on the combined ratings of all the KSs, are then used to construct new partial sentence hypotheses. The blackboard is then re-evaluated to find the most promising sentence hypothesis; this hypothesis then becomes the focal point for the next hypothesize-and-test cycle.

A mediator module (the "recognition overlord") is responsible for maintaining the blackboard, calculating combined ratings from the ratings assigned to hypotheses by the individual KSs, and deciding when to stop and accept a solution (or give up). The rating of the sentence hypotheses is the mechanism for attention focusing. A best-first strategy is used—the currently highest rated hypothesis is the one used as the context for the next cycle. If an error is made, the rating of the incorrect hypothesis will, hopefully, eventually degrade and attention will be focused to the sentence hypothesis which now has the highest rating.

Hearsayl contains three KSs:

1) The *acoustic-phonetic* KS deals with the sounds of the words of the input language and how they relate to the speech signal. It obtains (from a preprocessing module called *EAR*) a representation of the speech signal as an errorful (or course!) sequence of segments, each segment being labeled with a phonetic-like label. The input language is specified to the KS as a lexicon of words in which each word is "spelled" as a sequence of phonemic symbols (with some alternative spellings). The KS both hypothesizes words (from the segments) and evaluates the word hypotheses of other KSs.

2) The *syntax* KS deals with the orderings of words in the utterance according to the specified grammar of the input language. This grammar is specified to the KS in BNF notation. Given some contiguous word hypotheses, the KS can evaluate them for consistency with the grammar and also can hypothesize additional words which are likely to occur contiguous to them.

3) The *semantics* KS deals with the meaning of words and phrases of the input language, in the context of the task. Only one task semantics KS has been programmed (for "Voice-chess"—playing a game of chess verbally); its design is highly explicit to the one task. This KS hypothesizes and rates sentences and portions of sentences based on the chess moves they represent; it uses both the legality of the move in the current chess board position as well as the "goodness" of the move (as determined by a chess-playing program which the KS consults).

Hearsayl Performance

The Hearsayl system first demonstrated live, connected-speech recognition in June, 1972, at a workshop held at CMU. Since that time, about two person-years have been spent in studying it and in

improving its performance. The system has been formally tested on a set of 144 connected speech utterances, containing 676 word tokens, spoken by five speakers, and consisting of four tasks (only one of which has had a semantics component programmed), with vocabularies ranging from 28 to 76 words. The system locates and correctly identifies about 93% of the words, using all three of its KSs. Without the use of the semantics KS, the accuracy decreases to 70%. It decreases further to about 30% when neither syntax nor semantics are used. Hearsayl operates in about 7 to 10 times real-time on a PDP10-KA10 (0.3 million instructions/sec. machine), using about 120K words (36-bits/word) for storage and programs.

Hearsayl Design Limitations

There are four major design decisions in the Hearsayl implementation of knowledge representation and cooperation which make it difficult to directly extend Hearsayl to more ambitious performance goals.

The *first*, and most important, of these limiting decisions concerns the use of the hypothesize-and-test paradigm. As implemented in Hearsayl, the paradigm is exploited only at the word level. That is, the information content of any hypothesis in the blackboard is limited to a description at the word level. The addition of non-word level KSs (i.e., KSs cooperating via either sub-word levels, such as syllables or phones, or via supra-word levels, such as phrases or concepts) thus becomes cumbersome because this knowledge must somehow be related to hypothesizing and testing at the word level.

Secondly, Hearsayl constrains the hypothesize-and-test paradigm to operate in a lockstep control sequence. The effect of this decision is to limit parallelism of execution (and thus reduce effectiveness on a multi-processor configuration); this is because the time required to complete a hypothesize-and-test cycle is the maximum time required by any single hypothesizer KS plus the maximum time required by any single verifier (testing) KS. Another disadvantage of this control scheme is that the time increases for the system to refocus attention, because there is no provision for any communication of partial results among KSs. Thus, for example, a rejection of a particular option word by a KS will not be noticed until all the KSs have tested all the option words.

A *third* weakness in the Hearsayl implementation concerns the structure of the blackboard: there is *no* provision for specifying relationships among alternative sentence hypotheses. This absence has the effect of increasing the overall computation time and increasing the time to refocus attention, because the information gained by working on one hypothesis cannot be shared by propagating it to other relevant hypotheses.

A *fourth* limiting design decision relates to how a global problem-solving strategy is implemented in Hearsayl: The policies for attention-focusing and control are embedded in the recognition overlord module in an *ad hoc* fashion—there is no coherent structure for the algorithms and they are "wired in" to the kernel of the system, rather than being available for easy manipulation and experimentation. Thus it is awkward to modify and evaluate policy algorithms.

5. Overview of HearsayII

HearsayII represents the step following Hearsayl in the sequence of increasingly ambitious systems for speech understanding. The major changes to the system structure are a) in the representation of knowledge in the blackboard and b) in the manner of activation and attention-focusing of KSs.

The Blackboard of HearsayII

The blackboard has been extended and generalized to allow a) the representation of all levels of information (acoustic, phonetic, syllabic, etc.) in addition to the lexical and sentence levels of Hearsayl and b) the explicit representation of relationships among hypotheses.

The blackboard is partitioned into distinct information *levels*; each level is used to hold a different (and potentially complete) representation of the utterance. Associated with each level is a set of primitive elements appropriate for representing the problem at that level. (For example, the elements at the lexical level are the *words* of the vocabulary to be recognized, while the elements at the phonetic level are the *phones* of English.) Each hypothesis exists at a particular level and is labeled as being a particular element of the set of primitive elements at that level. The choice of levels (and the set of elements at each level) is not prespecified by the kernel of the system. To the kernel, all levels are uniform; so new ones can be added at any time. The configuration of levels that is currently in use is shown in Figure 2.[4]

Parametric Level—The parametric level holds the most basic representation of the utterance that the system has; it is the only direct input to the machine about the acoustic signal. Several different sets of parameters are being used in HearsayII interchangeably: 1/3-octave filter-band energies measured every 10 msec., LPC-derived vocal-tract parameters, and wide-band energies and zero-crossing counts.

4 An elaboration of the following description can be found in Shockey and Erman [74].

Segmental Level—This level represents the utterance as labeled acoustic segments. Although the set of labels is phonetic-like, the level is not intended to be phonetic—the segmentation and labeling reflect acoustic manifestation and do not, for example, attempt to compensate for the context of the segments or attempt to combine acoustically dissimilar segments into (phonetic) units.

Phonetic Level—At this level, the utterance is represented by a phonetic description. This is a *broad* phonetic description in that the size (duration) of the units is on the order of the "size" of phonemes; it is a *fine* phonetic description to the extent that each element is labeled with a fairly detailed allophonic classification (e.g., "stressed, nasalized [I]").

Surface-Phonemic Level—This level, named by seemingly contradicting terms, represents the utterance by phoneme-like units, with the addition of modifiers, such as stress and boundary (word, morpheme, syllable) markings.

Syllabic Level—The unit of representation here is the syllable.

Lexical Level—The unit of information at this level is the word.

Phrasal Level— Phrases appear at this level. In fact, since a level may contain arbitrarily many "sub-levels" of elements (using "links", as described below), traditional kinds of syntactic trees are directly represented here.

The decomposition of the blackboard into distinct levels of representation can also be thought of as an a priori framework of a *plan* for problem-solving. Each level is a generic stage in the plan. The goal at each level is to create and validate hypotheses at that level. For example, the goal at the phonetic level is a phonetic transcription of the utterance. The overall goal of the system is to create (using "links", as described below) the most plausible network of hypotheses that sufficiently covers the levels. 'Plausible and sufficient' here refer to the judgment of the KSs; 'covering the levels' means a network that connects hypotheses which describe the speech signal (at the parametric level) to hypotheses which describe the semantic content of the utterance (at the phrasal level).

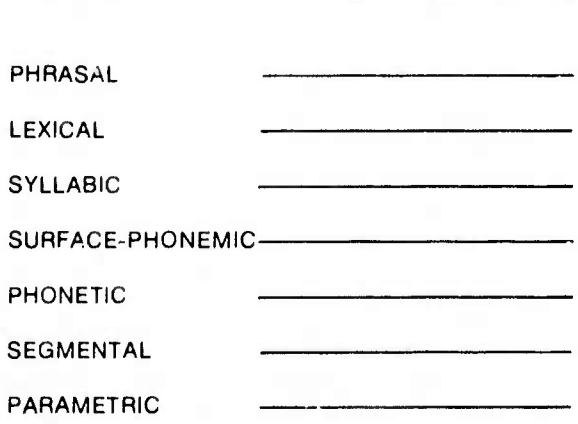


Figure 2: The Levels Currently Used in HearsayII.

The decomposition of the problem space into more levels than in HearsayI parallels the desire to decompose the KSs more finely, yielding more KSs, each of which is simpler and smaller. The principal resultant change in the configuration of KSs is that the single acoustic-phonetic KS of HearsayI is decomposed into about six KSs currently in HearsayII. For most KSs, the KS needs to deal with only one or two levels to apply its knowledge; it need not even be aware of the existence of other levels. Thus, each KS can be made as simple as its knowledge allows; its interface to the rest of the system is in units and concepts which are natural to it. Also, new levels can be added as new KSs are designed which need to use them. (For example, the syllabic level was a fairly late addition to the configuration—only two KSs needed to be modified when it was added.)

Activation of Knowledge Sources

A KS is instantiated as a knowledge-source process whenever the blackboard exhibits characteristics which satisfy a "precondition" of the KS. A *precondition* of a KS is a description of some partial state of the blackboard which defines when and where the KS can contribute its knowledge by modifying the blackboard. A KS carries out these actions with respect to a particular *context*, the context being some arbitrary subset of the previously generated hypotheses in the blackboard. Thus, new hypotheses or modifications to existing hypotheses are constructed from the (static) knowledge of the KS and the educated guesses made at some previous time by another KSs.

The modifications made by any given KS process are expected to trigger further KSs by creating new conditions in the blackboard to which those KSs, in turn, respond. The structure of a hypothesis is designed to allow the preconditions of most KSs to be sensitive to a single, simple change in some hypothesis (e.g., the creation of a new hypothesis of a particular type, a change of a rating, or the creation of a structural link between particular kinds of hypotheses). Through this *data-directed* interpretation of the hypothesize-and-test paradigm, KSs can

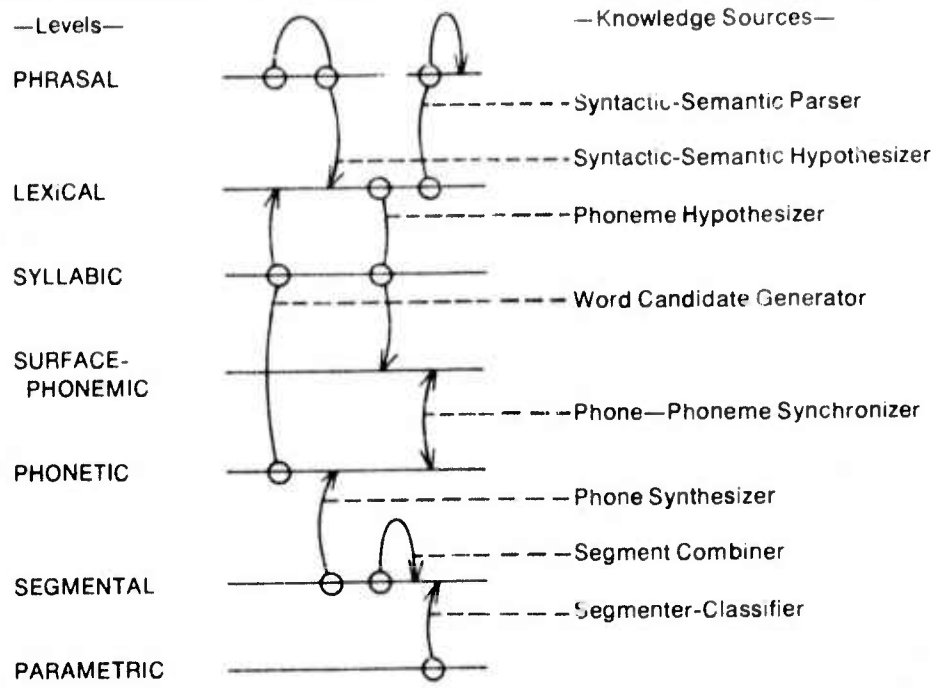


Figure 3: The Current Knowledge Sources in Hearsayll.

also exhibit a high degree of asynchronous activity and potential parallelism.[5]

As examples of KSs, Figure 3 shows many of the current set. The levels are indicated by horizontal lines in the figure and are labeled at the left. The KSs are indicated by arcs connecting levels; the starting point(s) of an arc indicates the level(s) of major "input" for the KS, and the end point indicates the "output" level where the KS's major actions occur. In general, the action of most of these particular KSs is to create links between hypotheses on its input level(s) and: 1) existing hypotheses on its output level, if appropriate ones are already there, or 2) hypotheses that it creates on its output level.

5 One might think of this model for data-directed activation of KSs as a production system (Newell [73]) which is executed asynchronously. The preconditions correspond to the left-hand sides (conditions) of productions, and the KSs correspond to the right-hand sides (actions) of the productions. Conceptually, these left-hand sides are evaluated continuously. When a precondition is satisfied, an instantiation of the corresponding right-hand side of its production is created; this instantiation is executed at some arbitrary subsequent time (perhaps subject to instantiation scheduling constraints).

The *Segmenter-Classifier* KS uses the parametric description of the speech signal to produce a labeled acoustic segmentation. (See Goldberg et al. [75] for a description of the algorithm used.) For any portion of the utterance, several possible alternative segmentations and labels may be produced.

The *Segment Combiner* combines similar adjacent segments into larger units. It is triggered on each new hypothesis at the segmental level.

The *Phone Synthesizer* uses labeled acoustic segments to generate elements at the phonetic level. This procedure is sometimes a fairly direct renaming of an hypothesis at the segmental level, perhaps using the context of adjacent segments. In other cases, phone synthesis requires the combining of several segments (e.g., the generation of [t] from a segment of silence followed by a segment of aspiration) or the insertion of phones not indicated directly by the segmentation (e.g., hypothesizing the existence of an [I] if a vowel seems velarized and there is no [I] in the neighborhood). This KS is triggered whenever a new hypothesis is created at the segmental level.

The *Word Candidate Generator* uses phonetic information (primarily just at stressed locations and other areas of high phonetic reliability) to

generate word hypotheses. This is accomplished in a two-stage process, with a stop at the syllabic level, from which lexical retrieval is more effective. (In fact, there are really two separate KSs here—one that goes from phones to syllables, and one that goes from syllables to words.)

The *Phoneme Hypothesizer* KS is activated whenever a word hypothesis is created (at the lexical level) which is not yet supported by hypotheses at the surface-phonemic level. Its action is to create one or more sequences at the surface-phonemic level which represent alternative pronunciations of the word. (These pronunciations are prespecified as entries in a dictionary.) It also creates the syllable hypotheses for the word, if they do not already exist.

The *Phone—Phoneme Synchronizer* is triggered whenever an hypothesis is created at either the phonetic or the surface-phonemic level. This KS attempts to link up the new hypothesis with hypotheses at the other level. This linking may be many-to-one in either direction.

The *Syntactic-Semantic Parser* uses the syntactic and semantic definition of the input language to build parses at the phrasal level. It is triggered by new word and phrasal hypotheses. This KS is not restricted to left-to-right parsing, but rather works piecemeal wherever hypotheses occur. One of its responsibilities is to identify possible interpretations for the entire utterance. (See Hayes-Roth and Mostow [75].)

The *Syntactic-Semantic Hypothesizer* also uses the syntactic and semantic definition of the input language. It hypothesizes phrasal and word hypotheses which are likely to occur adjacent to phrasal and word hypotheses already on the blackboard. This provides "top-down" activity in the system.

The *Rating Policy* KS operates at all levels of the blackboard. Its function is to propagate evaluations of hypotheses. For each hypothesis, this KS calculates ratings which are based on a) intrinsic ratings placed on the hypothesis by other KSs and b) the hypothesis' relationships to other hypotheses.

Hypotheses: Structure and Interrelationships

As described above, the structure of hypotheses at each level in the blackboard is identical (i.e., the interpretation of hypotheses at different levels is imposed by the KSs dealing with them.) The internal structure of an hypothesis consists of a fixed set of *attributes* (i.e., fields which are named); this set is the same for hypotheses at all levels of representation in the blackboard. The values of the attributes are set and modified by the KSs.

Besides holding information necessary to describe the hypothesis, attributes also serve as mechanisms for implementing the data-directed hypothesize-and-test paradigm. That is, a KS can specify particular attributes of hypotheses (usually at particular levels) which it wants to have monitored; whenever a change is made to one of these monitored attributes, the KS (through its precondition) can be activated and notified of the nature of the change.

Attributes can be grouped into several classes:

- The first class of attributes *names* the hypothesis. It contains the unique name of the hypothesis, the name of its level, and its label from the element set at that level.
- One very important set of attributes specifies *structural relationships* with other hypotheses, as described below.
- The next class of attributes is composed of parameters which *rate* the hypothesis. These include separate numerical ratings derived from a) *a priori* information about the hypothesis (usually placed on the hypothesis by its creator KS), and b) information derived from its relationships to other hypotheses.
- Another set of attributes contains information about KS *attention* to the hypothesis. These include suggestions (by KSs) of what type of further processing should occur. These suggestions are *goals*.
- For speech, *time* is a fundamental concept, so the Hearsayll system has a class of attributes for describing the begin- and end-time and the duration of the event which the hypothesis represents. These attributes include ways of explicitly representing fuzzy notions of the times. Besides its descriptive importance, the time attribute class is used to partition the blackboard for efficient access; e.g., a KS can retrieve hypotheses which overlap a particular time region. Using both time and level, a two-dimensional partitioning occurs.
- The capability for arbitrary *KS-specific* attributes is also included. This can be used by a KS to hold arbitrary information about the hypothesis; in this way a KS need not hold state information about the hypothesis internally across activations of the KS and allows, for example, the implementation of generator functions. If several KSs share knowledge of the name of one of these attributes, each of them can access and modify the attribute's value and thus communicate just as if it were a "standard" attribute;

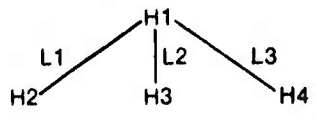
this can be used as an escape mechanism for explicit KS intercommunication.

—A unique class of hypothesis attributes, called *processing state* attributes, contains succinct summaries and classifications of the values of the other attributes. For example, the values of the rating attributes are summarized and the hypothesis is classified as either "unrated", "neutral" (noncommittal), "verified", "guaranteed" (strongly verified and unique), or "rejected". Other processing state attributes summarize the structural relationships with other hypotheses and characterize, for example, whether the hypothesis has been "sufficiently and consistently" described as an abstraction of hypotheses at lower levels. The processing state attributes are especially useful for efficiently triggering KSs; for example, a KS may specify in its precondition that it is to be activated whenever a hypothesis at a particular level becomes "verified". These attributes are also used for the goal-directed scheduling of KSs, as described in the next section.

Given a specific hypothesis, a KS can examine the value of any of its attributes. A KS source also needs the ability to retrieve sets of hypotheses whose attributes satisfy conditions in which the KS is interested; e.g., a KS may want to find all hypotheses at the phonetic level which are vowels and which occur within a particular time range. The system provides an *associative retrieval* search mechanism for accomplishing this. The search condition is specified by a *matching prototype* which is a partial specification of the components of a hypothesis.

Structural relationships between hypotheses in the blackboard are represented through the use of *links*; links provide a means of specifying contextual abstractions about the relationships of hypotheses. A link is an element of the blackboard which associates two hypotheses as an ordered pair; one of the nodes is termed the *upper hypothesis*, and the other is called the *lower hypothesis*. The lower hypothesis is said to *support* the upper hypothesis while the upper hypothesis is called a *use* of the lower one; in general, the lower hypothesis is at the same or a lower level in the blackboard than the upper hypothesis.

There are several types of links, with the types describing various kinds of relationships. Consider this structure:



H1 is the upper hypothesis and H2, H3, and H4 are the lower hypotheses of links L1, L2, and L3, respectively. If the links are all of type *OR*, the interpretation is that H1 is either an H2 or an H3 or an H4. This is one way that alternative descriptions are possible. If the links in the figure are of type *AND*, the interpretation is that all of the lower hypotheses are necessary to support the existence of H1. Variants of the AND- and OR-links are also used. An important one is the *SEQUENCE* link; it is similar to the AND-link except that a contiguous time-ordering is implied on the set of lower hypotheses supporting the upper hypothesis—if the links in the figure are SEQUENCE links, then H4 follows H3 which follows H2.

Besides showing structural relationships between hypotheses (e.g., that one hypothesis is composed of several other units), a link is a statement about the degree to which one hypothesis implies (i.e., "gives evidence for the existence of") another hypothesis. The strength of the implication is held as attributes of the link. The sense of the implication may be negative; that is, a link may indicate that one hypothesis is evidence for the *in*validity of another. This statement of implication may be bidirectional; the existence of the upper hypothesis may give credence to the existence of the lower hypothesis and vice versa. Finally, these relationships can be constructed in an iterative manner; links can be added between existing hypotheses by KSs as they discover new evidence for support.

Just as an hypothesis can have more than one lower link, so it can have several upper links. Each of these represents a different use of the hypothesis; the uses may be competing or complementary. The ability to have multiple uses and supports of the same hypothesis, as opposed to creating duplicates for each competing use and abstraction, serves to keep the blackboard compact and thereby reduces the combinatoric explosion in the search space. Further, since all the information about the hypothesis is localized, all uses and supports of the hypothesis automatically and immediately share any new information added to the hypothesis by any KSs. As changes are made to a hypothesis, some of its uses and supports may conflict with each other; if these conflicts become too large, a KS can decide to resolve them by either eliminating some of the conflicting attributes or by *splitting* the hypothesis into two or more hypotheses, each of which is more internally consistent.

Goal-Directed Scheduling of Knowledge Sources

As described earlier, the overall goal of the system is to create the most plausible network of hypotheses that sufficiently spans the levels. At any

instant of time, the blackboard may contain many incomplete networks, each of which is plausible as far as it goes. Some of these incomplete networks may also share subnetworks. Through KS activity, incomplete networks can be expanded (or contracted) and may be joined together (or fragmented). At any time, there may be many places in the blackboard which satisfy the (precondition) contexts for the activation of particular KSs. The task of *goal-directed scheduling* is that of deciding which of these sites should be allocated computing resources.

Several of the attribute classes of a hypothesis can be helpful in making scheduling decisions. Particularly valuable are the values of the attention attributes, which, as described earlier, are indicators telling how much computation has been expended on the hypothesis and suggestions by KSs of how desirable it is to devote further effort on the hypothesis (along with the kinds of processing that are desirable). The processing-state attributes and the ratings are also valuable for making scheduling decisions.

The implementation of the goal-directed scheduling strategy is separated from the actions of individual KSs. That is, the decision of whether a KS can contribute in a particular context is local to the KS, while the assignment of that KS to one of the many contexts on which it can possibly operate is made more globally. The three aspects of a) decoupling of focusing strategy from KS activity, b) decoupling of the data environment (blackboard) from the control flow (KS activation), and c) the limited context in which a KS operates, together permit a quick refocusing of attention of KSs. The ability to refocus quickly is very important because the errorful nature of the KS activity leads to many incomplete and possibly contradictory hypothesis networks; thus, as soon as possible after a network no longer seems promising, the resources of the system should be employed elsewhere.

Implementation and Current Status

Hearsay II is implemented (as was HearsayI) on the PDP10 in SAIL (VanLehn [73]), an extended Algol-60. A number of language mechanisms—particularly the flexible macro facility—are used to extend the language to include the kernel of the HearsayII system; the result is a problem-oriented programming system for writing KSs and exploring various configurations. The major facilities provided include:

- —KS definition facilities,
- —blackboard accessing routines—both direct and associative retrieval,
- —blackboard modification routines,
- —a scheduler which activates KSs,
- —an overlay facility which extends the 256K-word address space so that large configurations can be used,
- —blackboard monitoring and tracing facilities,
- —general-purpose tools for experimenter interaction with KSs, including breakpoints, execution tracing, examination and modification of variables, and execution of functions of the KS,
- —tools for building high-level debugging and interactive features that are KS-specific,
- —a package for graphical output of blackboard structures,
- —a timing package for determining execution costs, and
- —a means of reading "cliche" files—stored sequences of commands used for configuring and controlling the system.

The system that results is highly structured and has many conventions to which the participating researchers must adhere. This is necessary in order to maintain a system that many people are modifying and using concurrently. (There are currently about five people maintaining and modifying the kernel and approximately a dozen others experimenting with various KS configurations—a usable and up-to-date system must be operational at all times.)

The kernel has been operational since spring, 1974, and has gone through several major implementation iterations. All the KSs described above are operational; several of them represent second or third generation versions. Because the overlay facility has only just come up (summer, 1975), performance of the system as a whole is still unknown; the KSs have been developed using small configurations at a time. It is expected that preliminary over-all performance information will be available by the end of 1975, but development will continue over the foreseeable future—as long as progress continues to be made.

Although HearsayII is running on a uni-processor, it is implemented using multiple processes. The asynchronous nature of KS activation raises a number of issues related to interaction on the blackboard. In particular, because the execution of a KS may be delayed for an arbitrary period following the blackboard modification which triggered the KS, it is possible that intervening actions (of other KSs) may have invalidated its triggering conditions by the time that it actually executes. Mechanisms have been developed to handle these problems. This aspect of the research is described in Lesser, Fennell, Erman, and Reddy [74], Fennell [75], and Fennell and Lesser [75].

The HearsayII system also contains facilities for

simulating its execution on a multi-processor machine. Here the issues of process interference, resource locking (and process deadlock) and processor utilization are met. The papers referenced in the preceding paragraph also describe these aspects in detail. The simulations, using just a subset of the current KSs[6], indicate that Hearsayll can effectively utilize as many as twelve processors, with even more likely as the other KSs are added and as the scheduler is improved to reduce conflicts.

A preliminary implementation of the Hearsayll kernel has been carried out on C.mmp (CMU's multi-mini-processor). This has validated the multi-processing design of the system. This implementation has been accomplished using the L* system (Newell and Robertson [75]). Much of the further investigation of Hearsayll will take place in this context.

Acknowledgments

A significant portion of the CMU Computer Science Department has been involved in the Hearsay efforts—the author is only one of many.

- —Raj Reddy is responsible for a large measure of the ideas, energy, and vision of this work; without him, the project would not have existed.
- —Richard Fennell, Rick Hayes-Roth, Victor Lesser, Richard Neely, and Linda Shockey have been instrumental in the ideas and their exploration.
- —Allen Newell has provided guidance and encouragement.
- —Greg Gill deserves special mention for his outstanding contribution of programming the Hearsayll kernel (several times).
- —Without taking all the space needed to describe each of their contributions, we would like to acknowledge the efforts of all members of the CMU "speech group", as well as the entire computer science department, for contributing to a first-rate research environment.

I wish to thank Vic Lesser for considerable help with this paper.

6 Only a subset of five KSs was used for these simulations because a) the overhead of simulation is very high and b) when the simulations began many of the current KSs either did not exist or were too undeveloped to use.

References

Baker [75] Baker, J. K. Stochastic modeling as a means of automatic speech recognition. Doctoral Dissertation, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA, 1975.

Erman, Fennell, Lesser and Reddy [73] Erman, L. D., R. D. Fennell, V. R. Lesser, and D. R. Reddy. System organizations for speech understanding: Implications of network and multiprocessor computer architectures for AI. *Proc. 3rd Inter. Joint Conf. on Artificial Intel.*, Stanford, CA, 1973, 194-199.

Erman [74] Erman, L. D. An environment and system for machine understanding of connected speech. Doctoral Dissertation, Computer Science Dept., Stanford University; Technical Report, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA, 1974.

Erman [74b] Erman, L. D. (Ed.) *Contributed Papers of the IEEE Symposium on Speech Recognition*. April 15-19, 1974, Pittsburgh, Pa., IEEE Cat. No. 74CH0878-9AE. Many of these papers have been reprinted in a special issue of *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-23, 1 (1975).

Fennell [75] Fennell, R. D. Multiprocess software architecture for AI problem solving. Doctoral Dissertation, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA, 1975.

Fennell and Lesser [75] Fennell, R. D. and V. R. Lesser. Parallelism in AI problem solving: A case study of Hearsayll. Sagamore (NY) Computer Conf. on Parallel Processing, 1975.

Goldberg et al. [74] Goldberg, H. G., D. R. Reddy, and R. Suslick. Parameter independent machine segmentation and labeling. In Erman [74b], 106-111.

Hayes-Roth and Mostow [75] Hayes-Roth, F. and D. J. Mostow. An automatically compilable recognition network for structured patterns. *Proc. 4th Inter. Joint Conf. on Artificial Intel.*, Tbilisi, USSR, 1975.

Lesser, Fennell, Erman, and Reddy [74] Lesser, V. R., R. D. Fennell, L. D. Erman, and D. R. Reddy. Organization of the Hearsayll speech understanding system. In Erman [74b], 11-21. Also appeared in *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-23, 1, (1975), 11-23.

Lesser [75] Lesser, V. R. Parallel processing in speech understanding systems: A survey of design problems. In Reddy [75], 481-499.

Lowerre [75] Lowerre, B. T. Doctoral Dissertation (in preparation). Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA 1975.

Neely [73] Neely, R. B. On the use of syntax and semantics in a speech understanding system. Doctoral Dissertation, Stanford University; Technical Report, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA, 1973.

Newell et al [71] Newell, A., J. Barnett, J. Forgie, C. Green, D. Klatt, J. C. R. Licklider, J. Munson, R. Reddy, and W. Woods. *Speech Understanding Systems: Final Report of a Study Group*. Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA, 1971. Also Elsevier/North-Holland, Amsterdam, 1973.

Newell [73] Newell, A. Production systems: Models of control structures. In W. C. Chase (Ed.), *Visual Information Processing*, Academic Press, NY, 463-526.

Newell and Robertson [75] Newell, A. and G. Robertson. Some issues in programming multi-miniprocessors. *Behav. Res. Methods and Instr., 7*, 2, 75-86.

Reddy [66] Reddy, D. R. An approach to computer speech recognition by direct analysis of the speech wave. Doctoral Dissertation, AI Memo No. 43, Computer Science Dept., Stanford University, Stanford, CA, 1966.

Reddy and Vicens [68] Reddy, D. R. and Vicens, P. J. A procedure for segmentation of connected speech. *J. Audio Engr. Soc., 16*, 4 (1968).

Reddy, Erman, and Neely [70] Reddy, D. R., L. D. Erman, and R. B. Neely, The CMU speech recognition project. *Proc. IEEE System Sciences and Cybernetics Conf.*, Pittsburgh, PA, 1970.

Reddy [71] Reddy, D. R. Speech recognition: Prospects for the seventies. *Proc. IFIP 1971*, Ljubljana, Yugoslavia, Invited paper section I-5 to I-13.

Reddy, Erman, and Neely [72] Reddy, D. R., L. D. Erman, and R. B. Neely, A mechanistic model of speech perception. *Computer Science Research Review 1971-72*, Computer Science Dept., Carnegie-Mellon University, Pittsburgh, PA, 1972, 7-15.

Reddy, Erman, and Neely [73] Reddy, D. R., L. D. Erman, and R. B. Neely, A model and a system for machine recognition of speech. *IEEE Trans. Audio and Electroacoustics*, AU-21, 3, 1973, 229-238.

Reddy, Erman, Fennell, and Neely [73] Reddy, D. R., L. D. Erman, R. D. Fennell, and R. B. Neely, The Hearsay speech understanding system: An example of the recognition process. *Proc. 3rd Inter. Joint Conf. on Artificial Intel.*, Stanford, CA, 1973, 185-193.

Reddy [75] Reddy, D. R. (Ed.), *Speech Recognition: Invited Papers of the IEEE Symposium*. April 15-19, 1974, Pittsburgh, PA, Academic Press, NY, 1975.

Shockey and Erman [74] Shockey, L. and L. D. Erman. Sub-lexical levels in the Hearsayll speech understanding system. In Erman [74b], 208-210.

Simon [66] Simon, H. A. Scientific discovery and the psychology of problem solving. *Mind and Cosmos: Essays in Contemporary Science and Philosophy*, Series in Philosophy of Science, University of Pittsburgh, Pittsburgh, PA, (1966), 3, 22-40.

Simon [71] Simon, H. A. The theory of problem solving. In *Information Processing 71*, North-Holland, 1971, 261-277.

VanLehn [73] VanLehn, K. A. SAIL User Manual. Memo AIM-204, Stanford Artificial Intelligence Laboratory, Stanford University, Stanford, CA, 1973.

Vicens [69] Vicens, P. Aspects of speech recognition. Doctoral Dissertation, Report CS-127, Computer Science Dept., Stanford University, Stanford, CA, 1969.

A MULTI-LEVEL ORGANIZATION FOR PROBLEM SOLVING USING MANY, DIVERSE, COOPERATING SOURCES OF KNOWLEDGE

Lee D. Erman and Victor R. Lesser
Computer Science Department[1]
Carnegie-Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

An organization is presented for implementing solutions to knowledge-based AI problems. The hypothesize-and-test paradigm is used as the basis for cooperation among many diverse and independent knowledge sources (KS's). The KS's are assumed individually to be errorful and incomplete.

A uniform and integrated multi-level structure, the *blackboard*, holds the current state of the system. Knowledge sources cooperate by creating, accessing, and modifying elements in the blackboard. The activation of a KS is data-driven, based on the occurrence of patterns in the blackboard which match templates specified by the knowledge source.

Each level in the blackboard specifies a different representation of the problem space; the sequence of levels forms a loose hierarchy in which the elements at each level can approximately be described as abstractions of elements at the next lower level. This decomposition can be thought of as an *a priori* framework of a plan for solving the problem; each level is a generic stage in the plan.

The elements at each level in the blackboard are hypotheses about some aspect of that level. The internal structure of an hypothesis consists of a fixed set of attributes; this set is the same for hypotheses at all levels of representation in the blackboard. These attributes are selected to serve as mechanisms for implementing the data-directed hypothesize-and-test paradigm and for efficient goal-directed scheduling of KS's. Knowledge sources may create networks of structural relationships among hypotheses. These relationships, which are explicit in the blackboard, serve to represent inferences and deductions made by the KS's about the hypotheses; they also allow competing and overlapping partial solutions to be handled in an integrated manner.

The HearsayII speech-understanding system is an implementation of this organization; it is used here as an example for descriptive purposes.

INTRODUCTION

This paper describes an organization for knowledge-based artificial intelligence (AI) programs. Although this organization has been derived while developing several generations of speech understanding systems, we feel that it has general application to other domains of large AI problems (e.g., vision,[2] robotics, chess, natural language understanding, and protocol analysis).

Our efforts follow from the early work of Reddy (1966) and Reddy and Vicens (Vicens, 1969), through the Hearsay1 system (Reddy, et al., 1973a, 1973b; Erman, 1974), which was the first demonstrable connected-speech understanding system, up through the currently developing HearsayII system (Erman, et al., 1973; Lesser, et al., 1974; Fennell, 1975).[3] These efforts have increasingly focused on the overall system organization for solving the problem; this has resulted in the design and construction of a sophisticated and structured environment within which problem-solving strategies are developed. Others working in this area also consider this aspect important.[1] The HearsayII system will be used here as the primary example for describing the organization.

THE PROBLEM

The class of AI problem that is addressed in this paper is characterized by having a large problem space and the requirement of a large amount of knowledge for its solution. The large amount of explicit knowledge differentiates these problems from other AI areas (e.g., theorem-proving) in which very general "weak" methods are applied using meager amounts of built-in knowledge (Newell, 1969). Further, the knowledge needed covers a wide and diverse set of areas (some examples in the speech understanding problem are signal analysis, acoustic-phonetics, phonology, syntax, semantics, and pragmatics). We call each such area a *knowledge-source* (KS) and also define a KS to be an agent which embodies the knowledge of its area and which can take actions based on that knowledge.[2]

The sources of knowledge are often incomplete and approximate. This errorful nature may be traced to three sources: First, the *theory* on which the KS is based may be incomplete or incorrect. For example, modern phonological theories, as applied to the speech problem, are often vague and incomplete. Second, the *implementation* of a KS may be incomplete or incorrect; this may be caused by an incorrect translation of the theory to the program or by an intentionally heuristic implementation of the theory. Finally, the knowledge source may be operating on incorrect or incomplete *data* supplied to it by other KS's.[3]

As one knowledge source makes errors and creates ambiguities, other KS's must be brought to bear to correct and clarify those actions. This KS cooperation should occur as soon as possible after the introduction of an error or ambiguity in order to limit its ramifications.

A mechanism for providing this high degree of cooperation is the *hypothesize-and-test* paradigm. In this paradigm, solution-finding is viewed as an iterative process. Each step in the iteration involves a) the creation of an hypothesis, which is an "educated guess" about some aspect of

1 This research was supported in part by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

2 Reddy (1973) is a comparison of the speech and vision problem domains.

3 Lesser et al (1975) contains a detailed description of HearsayII, including the design decisions which mark its derivation from Hearsay1.

1 Newell, et al., (1971) contains an excellent in-depth study of the speech understanding problem. The current state-of-the-art is represented in the papers of the 1974 IEEE Symposium on Speech Recognition (Erman, 1974b; Reddy, 1975). In particular, Barnett (1973, 1975), and Rovner, et al., (1974) also describe highly structured systems; Baker (1974) has a highly structured system based on a simple Markov model.

2 For the purposes of this discussion, a KS can be considered static; i.e., whether a KS learns from experience is an issue that is orthogonal to this organization.

3 This may also include externally supplied data (e.g., the digitized acoustic wave-form which is the input to the speech-understanding system); the transducers of these data can be considered to be KS's which also introduce error.

the problem, and b) a test of the plausibility of the hypothesis. Both of these steps use a priori knowledge about the problem, as well as the previously generated hypotheses. This iterative guess-building terminates when a consistent hypothesis is generated which satisfies the requirements of an overall solution.

As a strategy for developing such systems, one needs the ability to add and replace sources of knowledge and to explore different control strategies. Thus, such changes must be relatively easy to accomplish; there must also be ways to evaluate the performance of the system in general and the roles of the various knowledge sources and control strategies in particular. This ability to experiment conveniently with the system is crucial if the amount of knowledge is large and many people are needed to introduce and validate it. One means of helping to provide these flexibilities is to require that KS's be independent.

Because the problems are large and require many computation steps for their solution, the system must be efficient in its computation. This must be certainly true for a "production" application system; however, it must also be reasonably efficient in the development versions because of the experimental way that a complex, knowledge-based system is developed. That is, many iterative runs over a significant amount of test data must be made to develop and evaluate the knowledge sources and control strategies.

MODEL FOR COOPERATION OF KNOWLEDGE SOURCES

The requirement that knowledge sources be independent implies that the functioning (and very existence) of each must not be necessary or crucial to the others. On the other hand, the KS's are required to cooperate in the iterative guess-building, using and correcting one another's guesses; this implies that there must be interaction among the processes. These two opposing requirements have led to a design in which each KS interfaces to the others externally in a uniform way that is identical across KS's and in which no knowledge source knows what or how many other KS's exist. The interface is implemented as a dynamic global data structure, called the *blackboard*. [1] The primary units in the blackboard are guesses about particular aspects of the problem; these units, which have a uniform structure throughout the blackboard, are called *hypotheses*. At any time, the blackboard holds the current state of the system; it contains all the guesses about the problem that exist. Subsets of hypotheses represent partial solutions to the entire problem; these may compete with the partial solutions represented by other (perhaps overlapping) subsets.

Each knowledge source may access any information in the blackboard. Each may add information to the blackboard by creating (or deleting) hypotheses, by modifying existing hypotheses, and by establishing or modifying explicit structural relationships among hypotheses. The generation and modification of globally accessible hypotheses is the exclusive means of communication among the diverse KS's. This mechanism of cooperation, which is an implementation of the hypothesize-and-test paradigm, allows a KS to contribute knowledge without being aware of which other KS's will use the information or which KS supplied the information that it used. It is in this way that knowledge sources are made independent and separable. The structural relationships (which are mentioned above and which will be described below) form a network of the hypotheses and are used to represent the deductions and inferences which caused a KS to generate one hypothesis from others. The explicit retention in the blackboard of these dependency relationships is used to hold, among other things, competing hypotheses. Because these are held in an integrated manner, selective backtracking for error recovery and other search strategies can be implemented in an efficient and non-redundant way.

Decomposition of Knowledge

The decomposition of the overall task into various knowledge sources is regarded as being *natural*; i.e., the units of the decomposition represent those pieces of knowledge which can be distinguished and recognized as being somehow naturally independent.[1] Such a scheme of "inverse decomposition" (or, composition) seems very natural for many problem-solving tasks, and it fits well into the hypothesize-and-test approach to problem-solving. As long as a sufficient "covering set" of knowledge areas required for problem solution is maintained, one can freely add new knowledge sources, or replace or delete old ones. Each knowledge source is self-contained, but each is expected to cooperate with the other knowledge sources that happen to be present in the system at that time.

A knowledge source is specified in three parts: a) the conditions under which it is to be activated (in terms of the conditions in the blackboard in which it is interested), b) the kinds of changes it makes to the blackboard, and 3) a procedural statement (program) of the algorithm which accomplishes those changes. A knowledge source is thus defined as possessing some processing capability which is able to solve some subproblem, given appropriate circumstances for its activation.

Activation of Knowledge Sources

A knowledge source is instantiated as a knowledge-source process whenever the blackboard exhibits characteristics which satisfy a "precondition" of the knowledge source. A *precondition* of a KS is a description of some partial state of the blackboard which defines when and where the KS can contribute its knowledge by modifying the blackboard. The KS contributes its knowledge through the mechanism of making hypotheses and evaluating and modifying the contributions of other knowledge sources (by verifying and rating or rejecting the hypotheses made by other knowledge sources). A KS carries out these actions with respect to a particular *context*, the context being some arbitrary subset of the previously generated hypotheses in the blackboard. Thus, new hypotheses or modifications to existing hypotheses are constructed from the (static) knowledge of the KS and the educated guesses made at some previous time by other knowledge sources.

The modifications made by any given knowledge-source process are expected to trigger further knowledge sources by creating new conditions in the blackboard to which those knowledge sources, in turn, respond. The structure of a hypothesis is so designed as to allow the preconditions of most KS's to be sensitive to a single, simple change in some hypothesis (such as the creation of a new hypothesis of a particular type, a change of a rating, or the creation of a structural link between particular kinds of hypotheses). Through this data-directed interpretation of the hypothesize-

---

[1] The term "blackboard" was used by Simon (1966) in describing a mechanism in long-term memory as part of a theory of the psychology of problem-solving. Simon (1971) further develops this concept and elaborates its uses in the context of an abstract model for problem-solving.

[1] The approach taken in knowledge source decomposition is not an attempt to characterize somehow the overall problem solution process and then apply some sort of traffic flow analysis to its internal workings in order to decompose the total process into minimally interacting knowledge sources. Rather, knowledge sources are defined by starting with some intuitive notion about the various pieces of knowledge which could be incorporated in a useful way to help achieve a solution.

and-test paradigm, KS's can also exhibit a high degree of asynchronous activity and potential parallelism.[1]

Control schemes in which one KS explicitly invokes other KS's are not appropriate because of the requirement that KS's be independent and because the invocation of a KS may depend on a complex set of conditions which is created by the combined actions of several KS's. Further, such direct-calling schemes complicate KS's by requiring that they contain information about the KS's that they will call. These same arguments apply against a centralized control scheme which is explicitly predefined ("wired-in") for a particular set of KS's.

Decomposition of the Blackboard

The blackboard is partitioned into distinct information *levels*; each level is used to hold a different representation of the problem space. (Examples of levels in the speech problem are "syntactic", "lexical", "phonetic", and "acoustic"; examples in scene analysis are "picture point", "line segment", "region", and "object".) Associated with each level is a set of primitive elements appropriate for representing the problem at that level. (In the speech system, for example, the elements at the lexical level are the words of the vocabulary to be recognized, while the elements at the phonetic level are the phones (sounds) of English.) Each hypothesis exists at a particular level and is labeled as being a particular element of the set of primitive elements at that level.

The decomposition of the problem space into levels is a natural parallel to the decomposition into KS's of the knowledge that is to be brought to bear. For many KS's, the KS needs to deal with only one or a few levels to apply its knowledge; it need not even be aware of the existence of other levels. Thus, each KS can be made as simple as its knowledge allows; its interface to the rest of the system is in units and concepts which are natural to it. Also, new levels can be added as new sources of knowledge are designed which need to use them. Finally, it will be shown that the multi-level representation allows for efficiently sequencing the activity of the KS's in a non-deterministic manner and for making use of multiprocessing.

The sequence of levels forms a loose hierarchical structure in which the elements at each level can approximately be described as abstractions of elements at the next lower level.[2] (For example, an utterance is composed of phrases, which are made of words, put together as syllables, each of which can be described as a sequence of phones, each of which is composed of acoustic segments, each of which can be described by a sequence of ten-millisecond intervals with certain kinds of acoustic characteristics.)

Most of the relationships of a hypothesis are with hypotheses at its level or adjacent levels; further, these relationships can usually be derived (by a KS appropriate to the level) without having to delve below the level of abstraction of the hypothesis. This locality of context simplifies the function of knowledge sources. (Or from the other point of view, the decomposition of knowledge into sufficiently simple-acting KS's also simplifies and localizes relationships in the blackboard.)[1]

The decomposition of the blackboard into distinct levels of representation can also be thought of as an a priori framework of a *plan* for problem-solving. Each level is a generic stage in the plan. The goal at each level is to create and validate hypotheses at that level. The overall goal of the system is to create the most plausible network of hypotheses that sufficiently covers the levels. ('Plausible' and 'sufficiently' here mean "plausible and sufficient in the judgment of the knowledge sources".) In speech understanding, for example, the goal at the phonetic level is a phonetic transcription of the utterance, while the overall goal is a network which connects hypotheses directly derived from the acoustic input to hypotheses which describe the semantic content of the utterance.

The creation or modification of an hypothesis which is based on a context of hypotheses at a lower level (or levels) can be considered an action of *synthesis*, or abstraction; conversely, manipulations of an hypothesis based on a higher level context can be considered *analysis*, or elaboration. In order to reduce the propagation and accumulation of errors caused by KS's, both kinds of action are needed in the system.[2]

Because of the choice of decomposition, the context for an analysis or synthesis action is usually localized to the level just above or below the level at which the action takes place. However, this is not a requirement; in fact, an action which skips over several levels can serve strongly to direct the activity of the system and thereby significantly prune the search space. Such a jump over levels is equivalent to constructing a major step in a plan. Further, there is no requirement that a jump necessarily be filled in completely (or even partially) if KS's are confident enough in the consistency of the larger step. Thus, the KS's can dynamically define the granularity in the hypothesis network necessary to assure the desired degree of consistency; this granularity may vary at different places in the blackboard, depending on the particular structures that occur.

Appendix A contains a description of the blackboard and KS decompositions for the HearsayII speech-understanding system.

Hypotheses: Structure and Interrelationships

The internal structure of an hypothesis consists of a fixed set of attributes (named fields); this set is the same for hypotheses at all levels of representation in the blackboard. These attributes are selected to serve as mechanisms for implementing the data-directed hypothesize-and-test paradigm.[3] The values of the attributes are defined and modified by the KS's.

Attributes can be grouped into several classes:

1 One might think of this model for data-directed activation of KS's as a production system (Newell, 1973) which is executed asynchronously. The preconditions correspond to the left-hand sides (conditions) of productions, and the knowledge sources correspond to the right-hand sides (actions) of the productions. Conceptually, these left-hand sides are evaluated continuously. When a precondition is satisfied, an instantiation of the corresponding right-hand side of its production is created; this instantiation is executed at some arbitrary subsequent time (perhaps subject to instantiation scheduling constraints). It is interesting to note that this generalized form of hypothesize-and-test leads to a system organization with some characteristics also similar to QA4 (Rulifson, *et al.*, 1973) and PLANNER (Hewitt, 1972). In particular, there are strong similarities in the data-directed sequencing of processes.

2 Many of the ideas here fit neatly into Simon's description of a "nearly decomposable hierarchical system" (Simon, 1962).

1 This simplification of form and interaction is an expected characteristic of a nearly decomposable hierarchical system (ibid.).

2 The use of the terms 'analysis' and 'synthesis' here are reversed from their usual uses in the speech recognition domain. Traditionally, 'synthesis' means going from a higher-level representation (e.g., lexical) to the speech signal, while analysis refers to the other direction. In speech recognition, however, the objective is the synthesis of a meaning for the utterance from the pieces of data which make up the speech signal.

3 In HearsayII, a KS can specify particular attributes of hypotheses at particular levels which it wants to have monitored. Whenever a change is made to one of these monitored attributes, the KS can be activated and notified of the nature of the change. The section below on "Data-Directed Activation of Knowledge Sources" contains a more complete description of this process.

The first class of attributes *names* the hypothesis: it contains the unique name of the hypothesis, the name of its level, and its label from the element set at that level.

The next class of attributes is composed of parameters which *rate* the hypothesis. These include separate numerical ratings derived from a) a priori information about the hypothesis, b) analysis actions performed on the hypothesis, c) synthesis actions, and d) combinations of (a), (b), and (c).

Another set of attributes contains information about KS *attention* to the hypothesis. These include a cumulative measure of the amount of computation that has already been expended on the hypothesis as well as suggestions for how much more processing should occur and of what type (e.g., a general request for analysis or synthesis or a specific request for a change to some attributes. These suggestions are *goals*.).

One very important set of attributes describes the *structural relationships* with other hypotheses, as described below.

For each problem domain, it is likely that there are other attributes which are basic to the problem and which should be provided in the structure of the hypotheses; these form a *problem-specific* class of attributes. In speech understanding, for instance, time is a fundamental concept, so the HearsayII system has a class of attributes for describing the begin- and end-time and the duration of the event which the hypothesis represents. (These attributes include ways of explicitly representing fuzzy notions of the times.) For vision, likely attributes would include the location and dimension of the element and trajectory information for moving objects.

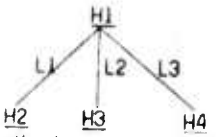
The capability for arbitrary *KS-specific* attributes is also included. This can be used by a KS to hold arbitrary information about the hypothesis; in this way a KS need not hold state information about the hypothesis across activations of the KS and allows, for example, the easy implementation of generator functions. If several KS's share knowledge of the name of one of these attributes, each of them can access and modify the attribute's value and thus communicate just as if it were a "standard" attribute; this can be used as an escape mechanism for explicit KS intercommunication.

A unique class of hypothesis attributes, called *processing state* attributes, contains succinct summaries and classifications of the values of the other attributes. For example, the values of the rating attributes are summarized and the hypothesis is classified as either "unrated", "neutral" (noncommittal), "verified", "guaranteed" (strongly verified and unique), or "rejected". Other processing state attributes summarize the structural relationships with other hypotheses and characterize, for example, whether the hypothesis has been "sufficiently and consistently" described synthetically (i.e., as an abstraction of hypotheses at lower levels). The processing state attributes are especially useful for efficiently triggering knowledge sources; for example, a KS may specify in its precondition that it is to be activated whenever a hypothesis at a particular level becomes "verified". These attributes are also used for the goal-directed scheduling of knowledge sources, as described in the next section.

Given a specific hypothesis, a KS can examine the value of any of its attributes. A knowledge source also needs the ability to retrieve sets of hypotheses whose attributes satisfy conditions in which the KS is interested. (E.g., a KS in the speech system may want to find all hypotheses at the phonetic level which are vowels and which occur within a particular time range.) The system provides an *associative retrieval* search mechanism for accomplishing this. The search condition is specified by a *matching-prototype*, which is a partial specification of the components of a hypothesis. This partial specification permits a component to be characterized by: a) a set of desired values or b) a don't-care condition. A matching-prototype is applied to a set of hypotheses;[1] those hypotheses whose component values match those specified by the matching-prototype are returned as the result of the search. (Associative retrieval of structural relationships among hypotheses is also provided.) More complex retrievals can be accomplished by combining the retrieval primitives in appropriate ways.

Structural relationships between nodes (hypotheses) in the blackboard are represented through the use of *links*; links provide a means of specifying contextual abstractions about the relationships of hypotheses. A link is an element which associates two hypotheses as an ordered pair; one of the nodes is termed the *upper hypothesis*, and the other is called the *lower hypothesis*. The lower hypothesis is said to *support* the upper hypothesis while the upper hypothesis is called a *use* of the lower one; in general, the lower hypothesis is at the same or a lower level in the blackboard than the upper hypothesis.

There are several types of links, with the types describing various kinds of relationships.[2] Consider this structure:



H1 is the upper hypothesis and H2, H3, and H4 are the lower hypotheses of links L1, L2, and L3, respectively. If the links are all of type *OR*, the interpretation is that H1 is either an H2 or an H3 or an H4. This is one way that alternative descriptions are possible. If the links in the figure are of type *AND*, the interpretation is that all of the lower hypotheses are necessary to support the existence of H1. (Note that, in general, all of the supporting (lower) links of a hypothesis are of the same type; one can thus talk of the "type of the hypothesis", which is the same as the type of all of its lower links.)

These two types of node represent different kinds of abstractions: the OR-node specifies a set/member relationship while the AND-node defines a composition abstraction. Variants of the AND- and OR-links are also possible. For example, a *SEQUENCE* link is similar to the AND-link except that an ordering is implied on the set of lower hypotheses supporting the upper hypothesis. (For the HearsayII speech understanding system, this ordering usually is interpreted as indicating a time ordering of the lower hypotheses.)

Besides showing analysis and synthesis relationships between hypotheses (e.g., that one hypothesis is composed of several other units), a link is a statement about the degree to which one hypothesis implies (i.e., "gives evidence for the existence of") another hypothesis. The strength of the implication is held as attributes of the link. The sense of the implication may be negative; that is, a link may indicate that one

---

1 This set can be derived by the KS from several sources. The HearsayII implementation includes the following primitive sources: a) all hypotheses (in the blackboard), b) all hypotheses at a particular level, c) all hypotheses at a particular level whose time attributes overlap a given interval (this provides an extremely efficient, two-dimension partition of the blackboard), and d) all hypotheses whose attributes which are being monitored (for the KS) have changed.

2 The particular kinds of relationships described here are some of those that were were designed for the speech problem. Although they undoubtedly are not the complete set for all conceivable needs, they do represent the kinds of relationships that need to be and are expressable in the blackboard.

hypothesis is evidence for the invalidity of another. This statement of implication may be bi-directional; the existence of the upper hypothesis may give credence to the existence of the lower hypothesis and *vice versa*. Finally, these relationships can be constructed in an iterative manner; links can be added between existing hypotheses by KS's as they discover new evidence for support.

Just as an hypothesis can have more than one lower link, so it can have several upper links. Each of these represents a different use of the hypothesis; the uses may be competing or complementary. The ability to have multiple uses and supports of the same hypothesis, as opposed to creating duplicates for each competing use and abstraction, serves to keep the blackboard compact and thereby reduces the combinatoric explosion in the search space. Further, since all the information about the hypothesis is localized, all uses and supports of the hypothesis automatically and immediately share any new information added to the hypothesis by any knowledge sources.

A problem with this localization can occur if the interactions between hypotheses span more than one level.[1] In this case, a particular support of the hypothesis (at a lower level) may be inconsistent with one (or more) of the uses of the hypothesis (at a higher level) but is consistent with other uses (or potential uses) of the hypothesis. In order to avoid duplicating the hypothesis, a mechanism, called a *connection matrix*, exists in the system. A connection matrix is an attribute of a hypothesis; its value specifies which of the alternative supports of the hypothesis are applicable ("connected to") which of its uses. The use of a connection matrix allows the results of previous decisions of KS's to be accumulated for future use and modification without necessitating contextual duplication of parts of the data base. This kind of reusage and multiple usage of blackboard structures, which results in localization of information, reduces much of the expensive backtracking that characterizes many problem-solving systems.

Appendix B contains an example of a structure built in the blackboard of the HearsayII system.

Goal-Directed Scheduling of Knowledge Sources

As described earlier, the overall goal of the system is to create the most plausible network of hypotheses that sufficiently spans the levels. At any instant of time, the blackboard may contain many incomplete networks, each of which is plausible as far as it goes. Some of these incomplete networks may also share subnetworks. Through the results of analysis and synthesis actions of knowledge sources, incomplete networks can be expanded (or contracted) and may be joined together (or fragmented). At any time, there may be many places in the blackboard which satisfy the (precondition) contexts for the activation of particular KS's. The task of *goal-directed scheduling* is to decide to which of these sites to allocate computing resources.

Several of the attribute classes of a hypothesis can be helpful in making scheduling decisions. Particularly valuable are the values of the attention attributes, which, as described earlier, are indicators telling how much computation has been expended on the hypotheses and suggestions by KS's of how desirable it is to devote further effort on the hypothesis (along with the kinds of processing that are desirable). The processing state attributes are also valuable for making scheduling decisions.

Using these kinds of information, a knowledge source might be scheduled for execution because it possesses the only processing capability available to be applied to an important incompletely explored area of the blackboard. For example, if the blackboard conlains focusing factors[2] which highlight activity in a blackboard region in which there are no structural connections between two adjoining levels, the scheduler should give a higher priority to a knowledge source which will attempt (as indicated in its external specifications) to make such a connection than to a knowledge source which is likely merely to perform a minor refinement on the ratings in one of the levels. However, if there are no such processes ready to execute, the scheduling algorithm can perform a type of means-ends analysis in which it schedules those knowledge sources which are likely to produce blackboard changes which, in turn, might trigger the activation of KS's in which the system is currently interested.

Another parameter for determining KS activation priority is the validity of the hypotheses which make up the context for the activation of the KS. This measurement can be used to implement a best-first strategy.

The implementation of the goal-directed scheduling strategy is separated from the actions of individual knowledge sources. That is, the decision of whether a KS can contribute in a particular context is local to the KS, while the assignment of that KS to one of the many contexts on which it can possibly operate is made more globally. The three aspects of a) decoupling of focusing strategy from knowledge-source activity, b) decoupling of the data environment (blackboard) from the control flow (KS activation), and c) the limited context in which a KS operates, together permit a quick refocusing of attention of KS's. The ability to refocus quickly is very important because the errorful nature of the KS activity leads to many incomplete and possibly contradictory hypothesis networks; thus, as soon as possible after a network no longer seems promising, the resources of the system should be employed elsewhere.[1]

IMPLEMENTATION OF DATA-DIRECTED KNOWLEDGE-SOURCE ACTIVATION

Associated with every knowledge source is a specification of the blackboard conditions required for the activation of that knowledge source. This specification, called a *precondition*, is a decision procedure whose tests are matching-prototypes and structural relationships which, when applied to the blackboard in an associative manner, detect the regions of the blackboard in which the knowledge source is interested. This procedure may contain arbitrarily complex decisions (based on current and past modifications to the blackboard) resulting in the activation of desired knowledge sources within the chosen contexts. The context corresponding to the discovered blackboard region which satisfies some knowledge source's precondition is used as an initial context in which to activate that knowledge source. The efficiency of the KS precondition evaluation is an important aspect of the system's implementation, especially as the knowledge is decomposed into more and smaller KS's and each KS activation requires less computation.

The HearsayII system, as an example of an implementation, makes precondition evaluation efficient by placing additional functions in the routines which modify the blackboard. These functions are activated whenever any KS modifies an attribute in the blackboard which some other KS has asked to be monitored. The essence of the modification is preserved in a data structure, called a *change set*, which is

1 Again, this fits well into Simon's formulation of hierarchical systems.

2 A focusing factor is a goal (attention attribute) attached to a hypothesis by a KS which indicates the kind of change desired in an attribute of the hypothesis and the desirability of the change. In addition, such goals may be specified for regions of the blackboard independent of the existence of hypotheses in the region.

1 The ideas of goal-directed scheduling are presented here only sketchily; Hayes-Roth et al (1975) provides a complete description of its use in the HearsayII system.

specific to the attribute changed and the KS which requested the monitoring. A KS specifies in a non-procedural way (either statically or dynamically) those attributes which it wants to monitor. In order to increase the efficiency, monitoring can further be localized to particular levels or even individual hypotheses.

Change sets serve to categorize blackboard modifications (events) and are thus useful in precondition evaluation since they limit the areas in the blackboard that need be examined in detail. As currently implemented in Hearsayll, the precondition evaluator of each knowledge source exists as a separale process which monitors changes in the data base (i.e., it monitors additions to those change sets in which the KS is interested). The precondition process is itself data-directed in that it is activated only when sufficient changes have been made in the blackboard (i.e., when an entry is made into one of its change sets, as a side-effect of a relevant blackboard modification). In effect, the precondition processes themselves have preconditions, albeit of a much simpler form than those possible for knowledge sources. For example, a precondition process in the speech system may specify that it should be activated whenever changes occur to two adjacent hypotheses at the word level or whenever support is added to the phrasal level. By using the (coarse) classifications afforded by change sets, the system avoids most unnecessary executions of the precondition processes. The major point is that the scheme of precondition evaluation is event-driven, being based on the occurrence of changes in the blackboard; i.e., it is only at points of modification to the blackboard that a precondition that was previously unsatisfied may become satisfied. In particular, precondition evaluators are not involved in a form of busy waiting in which they are constantly looking for something that is not yet there.

Once invoked, a precondition procedure uses sequences of associative retrievals and structural matches on portions of the blackboard in an attempt to establish a context satisfying the preconditions of one or more of "its" knowledge sources; any given precondition procedure may be responsible for instantiating several (related) knowledge sources. Notice that the data-directed nature of precondition evaluation and knowledge-source activation is linked closely to the primitive functions that are able to modify the data base, for it is only at points of modification that a precondition that was unsatisfied before may become satisfied. Hence, data base modification routines have the responsibility (although perhaps indirectly) of activating the precondition evaluation mechanism.

Implementation on Parallel Computers

Because of the independence of KS's and their data-directed activation, there is a great deal of potential parallelism in this organization. Trends in computer architecture indicate that large amounts of computing power will be economically realized in asynchronous multiprocessor networks. Thus, the implementation of such large AI programs on multiprocessors becomes an attractive goal. There are, however, a set of issues in such an implementation; most of these deal with interference among KS's when they attempt simultaneously to access the blackboard. Effective solutions to these problems have been developed in the HearsayII implementation; Lesser, *et al.*, (1974), Lesser (1975), Fennell (1975), and Fennell and Lesser (1975) describe these solutions.

ACKNOWLEDGMENTS

We wish to acknowledge the contributions of the following people: Richard Fennell for his major role in the development of Hearsayl and II (and, in particular, the multiprocessing aspects of the system organization) and for helpful suggestions for this paper, Raj Reddy for many of the basic ideas which have led to the organization described here Allen Newell for his insights into the task of problem-solving Gregory Gill for his untiring efforts in system implementation and Frederick Hayes-Roth for suggestions for this paper.

REFERENCES

Baker, James (*1974*), "The DRAGON System -- An Overview," in Erman (1974b).

Barnett, J. (*1973*), "A Vocal Data Management System," *IEEE Trans. Audio and Electroacoustics*, AU-21, 3.

Barnett, J. (*1975*, in press), "Module Linkage and Communication in Large Systems," in Reddy (1975).

Erman, L. D., R. D. Fennell, V. R. Lesser, and D. R. Reddy (*1973*), "System Organizations for Speech Understanding: Implications of Network and Multiprocessor Computer Architectures for AI," *Proc. 3rd Inter. Joint Conf. on Artificial Intel.*, Stanford, Ca., pp. 194-199.

Erman, L. D. (*1974*), "An Environment and System for Machine Understanding of Connected Speech," Ph.D. thesis, Comp. Sci. Dept, Stanford Univ.

Erman, L. D. (ed.) (*1974b*), *Contributed Papers of the IEEE Symposium on Speech Recognition*, April 15-19, 1974, Pittsburgh, Pa., IEEE Cat. No. 74CH0878-9AE. Many of these papers have been reprinted in *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-23, no. 1 (Feb., 1975).

Fennell, R. D., and V. R. Lesser (*1975*), "Parallelism in AI Problem-Solving: A Case Study of HearsayII," to be presented at the Sagamore (NY) Computer Conf. on Parallel Processing.

Fennell, R. D. (*1975*), Ph.D. thesis, Comp. Sci. Dept., Carnegie-Mellon Univ.

Hayes-Roth, F., V. R. Lesser, and D. W. Kosy (*1975*, in preparation), "A Design for Attentional Control in a Distributed-Logic Speech Understanding System," Tech. Report, Comp. Sci. Dept., Carnegie-Mellon Univ.

Hewitt, C. (*1972*), "Description and Theoretical Analysis (Using Schemata) of Planner: A Language for Proving Theorems and Manipulating Models in a Robot," AI Memo No. 251, MIT Project MAC.

Lesser, V. R., R. D. Fennell, L. D. Erman, & D. R. Reddy (*1974*), "Organization of the HEARSAY II Speech Understanding System," in Erman (1974b). Also appeared in *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-23, no. 1, pp. 11-23 (Feb., 1975).

Lesser, V. R. (*1975*, in press), "Parallel Processing in Speech Understanding Systems: A Survey of Design Problems," in Reddy (1975).

Newell, A. (*1969*), "Heuristic Programming: Ill-Structered Problems," in J. S. Aronofsky (ed.), *Progress in Operations Research*, 3, Wiley, 363-415.

Newell, A., J. Barnett, J. Forgie, C. Green, D. Klatt, J. C. R. Licklider, J. Munson, R. Reddy, and W. Woods (*1971*), *Speech Understanding Systems: Final Report of a Study Group*, pub. by North-Holland (1973).

Newell, A. (*1973*), "Production Systems: Models of Control Structures," in W. C. Chase (ed.) *Visual Information Processing*, Academic Press, pp. 463-526.

Reddy, D. R. (*1966*), "An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave," (Ph.D. thesis) AI Memo No. 43, Comp. Sci. Dept., Stanford Univ., Stanford, Ca.

Reddy, D. R., L. D. Erman, and R. B. Neely (*1973a*), "A Model and a System for Machine Recognition of Speech," *IEEE Trans. Audio and Electroacoustics*, AU-21, 3, pp. 229-238.

Reddy, D. R., L. D. Erman, R. D. Fennell, and R. B. Neely (*1973b*), "The HEARSAY Speech Understanding System: An Example of the Recognition Process," *Proc. 3rd Inter. Joint Conf. on Artificial Intel.*, Stanford, Ca., pp 185-193.

Reddy, D. R. (*1973*), "Eyes and Ears for Computers," Tech. Report, Comp. Sci. Dept., Carnegie-Mellon University, Keynote speech presented at Conf. on Cognitive Processes and Artifical Intelligence, Hamburg, April, 1973.

Reddy, R., and A. Newell (*1974*), "Knowledge and its Representation in a Speech Understanding System," in L. W. Gregg (ed.) *Knowledge and Cognition*, Lawrence Erlbaum Assoc., Washington, D. C., chap. 10.

Reddy, D. R. (ed.) (*1975*, in press), *Invited Papers of the IEEE Symposium on Speech Recognition*, April 15-19, 1974, Pittsburgh, Pa., Academic Press.

Rovner, P., Nash-Webber, B., and Woods, W. (*1974*), "Control Concepts in a Speech Understanding System," in Erman (1974b). Also appeared in *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-23, no. 1, pp. 136-140 (Feb., 1975).

Rulifson, J. F., et al. (*1973*), "QA4: A Procedural Calculus for Intuitive Reasoning," Technical Note 73, AI Center, Stanford Res. Inst.

Simon, H. A. (*1962*), "The Architecture of Complexity", *Proc. Amer. Philosophical Society*, 106, pp 467-482.

Simon, H. A. (*1966*), "Scientific Discovery and the Psychology of Problem Solving," *Mind and Cosmos: Essays in Contemporary Science and Philosophy*, in v. 3, Series in Philosophy of Science, Univ. of Pittsburgh, pp. 22-40.

Simon, H. A. (*1971*), "The Theory of Problem Solving," in *Information Processing 71*, North-Holland, pp. 261-277.

Vicens, P. (*1969*), "Aspects of Speech Recognition," Report CS-127, (Ph.D. Thesis), Comp. Sci. Dept., Stanford Univ.

IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-23, NO. 1, FEBRUARY 1975

Organization of the Hearsay II Speech Understanding System

VICTOR R. LESSER, RICHARD D. FENNELL, LEE D. ERMAN, AND D. RAJ REDDY,
MEMBER, IEEE

Abstract—Hearsay II (HSII) is a system currently under development at Carnegie-Mellon University to study the connected speech understanding problem. It is similar to Hearsay I (HSI) in that it is based on the hypothesize-and-test paradigm, using cooperating independent knowledge sources communicating with each other through a global data structure (blackboard). It differs in the sense that many of the limitations and shortcomings of HSI are resolved in HSII.

The main new features of the Hearsay II system structure are: 1) the representation of knowledge as self-activating, asynchronous, parallel processes, 2) the representation of the partial analysis in a generalized three-dimensional network (the dimensions being level of representation (e.g., acoustic, phonetic, phonemic, lexical, syntactic), time, and alternatives) with contextual and structural support connections explicitly specified, 3) a convenient modular structure for incorporating new knowledge into the system at any level, and 4) a system structure suitable for execution on a parallel processing system.

The main task domain under study is the retrieval of daily wire-service news stories upon voice request by the user. The main parametric representations used for this study are 1/3-octave filterbank and linear-predictive coding (LPC)-derived vocal tract parameters [10], [11]. The acoustic segmentation and labeling procedures are parameter-independent [7]. The acoustic, phonetic, and phonological components [23] are feature-based rewriting rules which transform the segmental units into higher level phonetic units. The vocabulary size for the task is approximately 1200 words. This vocabulary information is used to generate word-level hypotheses from phonetic and surface-phonemic levels based on prosodic (stress) information. The syntax for the task permits simple English-like sentences and is used to generate hypotheses based on the probability of occurrence of that grammatical construct [19]. The semantic model is based on the news items of the day, analysis of the conversation, and the presence of certain content words in the partial analysis. This knowledge is to be represented as a production system. The system is expected to be operational on a 16-processor minicomputer system [3] being built at Carnegie-Mellon University.

This paper deals primarily with the issues of the system organization of the HSII system.

INTRODUCTION

THE HEARSAY II (HSII) speech understanding system is a successor to the Hearsay I (HSI) system [16], [17]. HSII represents, in terms of both its system organization and its speech knowledge, a significant increase in sophistication and generality over HSI. The development of HSII has been based on two years of experience with a running version of HSI, a desire to exploit multiprocessor and network computer architecture for efficient implementation [3], [4], [6], and a desire to handle more complex speech task domains (e.g., larger vocabularies, less restricted grammars, and a more complete set of knowledge sources including prosodics, user models, etc.). While from a conceptual point of view HSII is a natural extension of the framework that HSI posited for a speech understanding system, it differs significantly in its design and in its details of implementation.[1]

THE HEARSAY SYSTEM MODEL

HSI was based on the view that the inherently errorful nature of connected speech processing could be handled only through the efficient use of multiple, diverse sources of knowledge [13], [15]. The major focus of the design of HSI was the development of a framework for representing these diverse sources of knowledge and their cooperation [18]. This framework is the conceptual legacy which forms the basis for the HSII design.

There are four dimensions along which knowledge representation in HSI can be described: 1) function, 2) structure, 3) cooperation, and 4) attention focusing.

The *function* of a knowledge source (KS) in HSI has three aspects. The first is for the KS to know when it has something useful to contribute, the second is to contribute its knowledge through the mechanism of making a *hypothesis* (guess) about some aspect of the speech utterance, and the third is to evaluate the contribution of other knowledge sources, i.e., to verify and reorder (or reject) the hypotheses made by other knowledge sources. Each of these aspects of a KS is carried out with respect to a particular context, the context being some subset of the previously generated hypotheses. Thus, new knowledge is built upon the educated guesses made at some previous time by other knowledge sources.

The *structure* of each knowledge source in HSI is specified so that it is independent and separable from all other KS's in the system. This permits the easy addition of new

Manuscript received March 29, 1974; revised August 15, 1974. This research was supported in part by the Defense Advanced Research Projects Agency under Contract F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research. This paper was presented at the IEEE Symposium on Speech Recognition, Carnegie-Mellon University, Pittsburgh, Pa., April 15–19, 1974.

The authors are with the Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa. 15213.

[1] Several other organizations for speech understanding systems have been implemented over the past few years. Included among the more interesting ones are the SDC system [2], [20], the Bolt Beranek and Newman (BBN) SPEECHLIS system [21], [24], the DRAGON system [1], and the IBM system [9].

 Annals No. 502AS002

types of KS's and replacement of KS's with alternative versions of those KS's. Thus, the system structure can be easily adapted to new speech task domains which have KS's specific to that domain, and the contribution of a particular KS to the total recognition effort can be more easily evaluated.

The choice of a framework for *cooperation* among KS's is intimately interwoven with the function and structure of knowledge in HSI. The mechanism for KS cooperation involves *hypothesizing* and *testing* (creating and evaluating) hypotheses in a *global data base* (blackboard). The generation and modification of globally accessible hypotheses thus becomes the primary means of communication between diverse KS's. This mechanism of cooperation allows a KS to contribute knowledge without being aware of which other KS's will use its knowledge or which KS contributed the knowledge that it used. Thus, each KS can be made independent and separable.

The global data base that KS's use for cooperation contains many possible interpretations of the speech data. Each of these interpretations represents a "limited" context in which a KS can possibly contribute information by proposing or validating hypotheses. *Attention focusing* of a KS involves choosing which of these limited contexts it will operate in and for how much processing time. The attention focusing strategy is decoupled from the functions of individual KS's. Thus, the decision of whether a KS can contribute in a particular context is local to the KS, while the assignment of that KS to one of the many contexts on which it can possibly operate is made more globally. This decoupling of focusing strategy from knowledge acquisition, together with the decoupling of the data environment (global data base) from control flow (KS invocation) and the limited context in which a KS operates, permits a quick refocusing of attention of KS's. The ability to refocus quickly is very important in a speech understanding system, because the errorful nature of the speech data and its processing leads to many potential interpretations of the speech. Thus, as soon as possible after an interpretation no longer seems the most promising, the activity of the system should be refocused to the new most promising interpretation.

OVERVIEW OF HSI

The following is a brief description of the HSI implementation for this model of knowledge source representation and cooperation. This description will then be used to contrast the differences of implementation philosophy between HSI and HSII. (More complete descriptions of HSI are contained in [5], [12], [16], [17].)

HSI Implementation Overview

The global data base of HSI consists of partial sentence hypotheses, each being a sequence of words with nonoverlapping time locations in the utterance. It is a *partial* sentence hypothesis because not all of the utterance need be described by the given sequence of words. In particular, gaps in the knowledge of the utterance are designated by "filler" words. The partial sentence hypotheses also contain confidence ratings for each word hypothesis and a composite rating for the overall sequence of words. A sentence hypothesis is the focal point that is used to invoke a knowledge source. The sentence hypothesis also contains the accumulation of all information that any KS has contributed to that hypothesis.

KS's are invoked in a lockstep sequence consisting of three phases: *poll*, *hypothesize*, and *test*. At each phase, all KS's are invoked for that phase, and the next phase does not commence until all KS's have completed the current one. The poll phase involves determining which KS's have something to contribute to the sentence hypothesis which is currently being focused upon; polling also determines how confident each KS is about its proposed contributions. The hypothesize phase consists of invoking the KS showing the most confidence about its proposed contribution of knowledge. This KS then hypothesizes a set of possible words (option words) for some (one) "filler" word in the speech utterance. The testing phase consists of each KS evaluating (verifying) the possible option words with respect to the given context. After all KS's have completed their verifications, the option words which seem most likely, based on the combined ratings of all the KS's, are then used to construct new partial sentence hypotheses. The global data base is then re-evaluated to find the most promising sentence hypothesis; this hypothesis then becomes the focal point for the next hypothesize-and-test cycle.

HSI Performance

The HSI system first demonstrated live, connected-speech recognition publicly in June 1972. Since that time, about three man-years have been spent in improving its performance. The system has been tested on a set of 144 connected speech utterances, containing 676 word tokens, spoken by five speakers, and consisting of four tasks, with vocabularies ranging from 28 to 76 words. On the average, the system locates and correctly identifies about 93 percent of the words, using all of its KS's. Without the use of the Semantics KS, the accuracy decreases to 70 percent. It decreases further to about 39 percent when neither Syntax nor Semantics are used. A more complete performance summary may be found in the Appendix.

HSI Design Limitations

There are four major design decisions in the HSI implementation of knowledge representation and cooperation which make it difficult to apply HSI to more complex speech tasks or multiprocessor environments.

The *first*, and most important, of these limiting decisions concerns the use of the hypothesize-and-test paradigm. As implemented in HSI, the paradigm is exploited only at the word level. The implication of hypothesizing and testing at only the word level is that the knowledge representation is uniform only with respect to cooperation at that level. That is, the information content of any

element in the global data base is limited to a description at the word level. The addition of nonword level KS's (i.e., KS's cooperating via either subword levels, such as syllables or phones, or via supraword levels, such as phrases or concepts) thus becomes cumbersome because this knowledge must somehow be related to hypothesizing and testing at the word level. This approach to nonword level KS's makes it difficult to add nonword knowledge and to evaluate the contribution of this knowledge. In addition, the inability to share nonword level information among KS's causes such information to be recomputed by each KS that needs it.

Second, HSI constrains the hypothesize-and-test paradigm to operate in a lockstep control sequence. The effect of this decision is to limit parallelism, because the time required to complete a hypothesize-and-test cycle is the maximum time required by any single hypothesizer KS plus the maximum time required by any single verifier (testing) KS. Another disadvantage of this control scheme is that it increases the time it takes the system to refocus attention, because there is no provision for any communication of partial results among KS's. Thus, for example, a rejection of a particular option word by a KS will not be noticed until all the KS's have tested all the option words.

The *third* weakness in the HSI implementation concerns the structure of the global data base: there is no provision for specifying relationships among alternative sentence hypotheses. The absence of relational structures among hypotheses has the effect of increasing the overall computation time and increasing the time to refocus attention, because the information gained by working on one hypothesis cannot be shared by propagating it to other relevant hypotheses.

The *fourth* limiting design decision relates to how a global problem-solving strategy (policy) is implemented in HSI: policy decisions, such as those involving attention focusing, are centralized (in a "Recognition Overlord"), and there is no coherent structure for the policy algorithms. The effect of having no explicit system structure for implementing policy decisions makes it very awkward to add or delete new policy algorithms and difficult to analyze the effectiveness of a policy and its interaction with other policies.

OVERVIEW OF HSII

Experience with HSI (as described above) has led to several important observations about a more general, uniform, and natural structure for representing and operating on the (dynamic) state of the utterance recognition.[2]

1) The information contained in hypotheses at different levels of knowledge representation may be encoded in essentially identical internal structures, except for the primitive unit of information held in an hypothesis. This structural homogeneity in the global data base allows the actions of hypothesizing and testing at these various levels to be treated in a uniform manner.

2) The different types of knowledge (and their relationships) present in speech may be naturally represented in a single, uniform data structure. This data structure is three-dimensional: one dimension represents information levels (e.g., phrasal, lexical, phonetic), the second represents speech time, and the third dimension contains alternative (competing) hypotheses at a particular level and time. These three dimensions form a convenient addressing structure for locating hypotheses.

3) There is a conceptually simple and uniform way of dynamically relating hypotheses at one level of knowledge to alternative hypotheses at that level and to hypotheses at other knowledge levels in the structure. The resulting structure is an AND/OR graph with modifications which provide for temporal relationships and selective dependency relationships.[3]

System Structure

The main goal of the HSII design is to extend the concepts developed in HSI for the representation and cooperation of knowledge at the word level to *all levels of knowledge* needed in a speech understanding system, based on the preceding observations.

The generalization of the hypothesize-and-test paradigm to all levels of speech knowledge implies the need for a mechanism for transferring information among levels. This mechanism is already embodied in the hypothesize-and-test paradigm; that is, one can characterize two types of hypothesization a knowledge source might be called upon to perform: horizontal and vertical hypothesization. A hypothesization is *horizontal* when a KS uses contextual information at a given knowledge level to predict new hypotheses at the same level (e.g., the hypothesization that the word "night" might follow the sequence of words "day"–"and"), whereas a hypothesization is *vertical* when a KS uses information at one level in the data base to predict new hypotheses at a different level (e.g., the generation of a hypothesis that a [T] occurred when a segment of silence is followed by a segment of aspiration).

The HSII implementation of the hypothesize-and-test paradigm has also resulted in a generalization of the lockstep control scheme for KS sequencing employed by HSI. HSII relaxes the constraints on the hypothesize-and-test paradigm and allows the KS processes to run in an asynchronous, data-directed manner. A KS is instantiated as a *KS process* whenever the data base exhibits characteristics which satisfy a "precondition" of the KS. A *precondition* of a KS is a description of some partial state of the data base which defines when and where the KS can contribute its knowledge by modifying the data base. Such a modification might be adding new hypotheses proposed by the KS (at the information level appropriate for that KS) or verifying (criticizing) hypotheses which already exist.

[2] The meaning of these observations will be made more clear by the further descriptions that follow.

[3] This latter feature refers to "connection matrices" and is described below in more detail.

The modifications made by any given KS process are expected to trigger further KS's by creating new conditions in the data base to which those KS's respond. The structure of a hypothesis has been so designed as to allow the preconditions of most KS's to be sensitive to a single, simple change in some hypothesis (such as the changing of a rating or the creation of a structural link). Through this data-directed interpretation of the hypothesize-and-test paradigm, HSII KS's exhibit a high degree of asynchrony and potential parallelism. A side-effect of this more general control scheme for HSII is that the overall problem-solving strategy need not be centralized and implemented as a monolithic overlord, but rather can be implemented as *policy modules* which operate in precisely the same manner as KS's.

The three-dimensional data base, augmented by the AND/OR structural relationships specified over that data base, permits information generated by one KS to be: 1) retained for use by other KS's, and 2) quickly propagated to other relevant parts of the data base. This retention and propagation provide two important features for solving a complex problem in which errors are highly likely. First, quick refocusing can occur when a particular path no longer appears promising. Second, "selective" backtracking may be used; i.e., when a KS finds that it has made an incorrect decision, it does not have to eliminate all information generated subsequent to that decision, but rather only that subset which depends on the incorrect decision. In this way, information generated by one knowledge source is retained and is usable by itself and other KS's in other relevant contexts.

Summarizing, HSII is based on the views: 1) that the state of the recognition can be represented in a uniform, multilevel data base, and 2) that speech knowledge can be characterized in a natural manner by describing many small KS's. These KS's react to certain states of the data base (via their preconditions) and, once instantiated as KS processes, provide their own changes to the data base which contribute to the progress of the recognition. The hypothesize-and-test paradigm, when stated in sufficiently nonrestrictive (parallel) terms, serves to describe the general interactions among these KS's. In particular, changes made by one or more KS processes may trigger other KS's to react to these changes by validating (testing) them or hypothesizing further changes. The intent of HSII is to provide a framework within which to explore various configurations of information levels, KS's, and global strategies.[4]

From a more general point of view, the goal of HSII is to provide a multiprocess-oriented software architecture to serve as a basis for systems of cooperating (but independent and asynchronous) data-directed KS processes. The purpose of such a structure is to achieve effective parallel search over a general artificial intelligence problem-solving graph, employing the hypothesize-and-test paradigm to generate the search graph and using a uniform, interconnected, multilevel global data base as the primary means of interprocess communication.

[4] It is interesting to note that this generalized form of hypothesize-and-test leads to a system organization with some characteristics similar to QA4 [22] and PLANNER [8]. In particular, there are strong similarities in the data-directed sequencing of processes.

HSII SYSTEM DESIGN AND IMPLEMENTATION

One can derive from the description of the desired HSII recognition process given above several basic components of the required system structure. First, a sufficiently general *structured global data base* is needed, through which the KS's may communicate by inserting hypotheses and by inspecting and modifying the hypotheses placed there by other KS's. Second, some means for describing the various KS's and their internal processing capabilities is required. Third, in order to have knowledge sources activated in a data-directed manner, a method is required by which a set of *preconditions* may be specified and associated with each KS. Fourth, in order to detect the satisfaction of these preconditions and in order to allow KS's to locate parts of the data base in which they are interested, two mechanisms are needed: 1) a *monitoring mechanism* to record where in the data base changes have occurred and the nature of those changes, and 2) an *associative retrieval mechanism* for accessing parts of the data base which conform to particular patterns specified by *matching prototypes*.

Elements of the System Structure

The following sections outline the HSII implementation of the various basic system components.

Global Data Base: The design of HSII is centered around a global data base (blackboard) which is accessible to all KS processes. The global data base is structured as a uniform, multilevel, interconnected data structure.

Each level in the data base contains a (potentially complete) representation of the utterance; the levels are differentiated by the units that make up the representation, e.g., phrases, words, phonemes. The system structure of HSII does not prespecify what the levels in the global data structure are to be. A particular configuration, called HSII-C0 (Configuration Zero), is being implemented as the first test of the HSII structure. Fig. 1 shows a schematic of the levels of HSII-C0. A more detailed description and justification for parts of this particular configuration can be found in [23]. This configuration will be used as the basis for examples to illustrate various aspects of the HSII system.

Parametric Level: The parametric level holds the most basic representation of the utterance that the system has; it is the only direct input to the machine about the acoustic signal. Several different sets of parameters are being used in HSII-C0 interchangeably: 1/3-octave filter-band energies measured every 10 ms, LPC-derived vocal-tract parameters [10], and wide-band energies and zero-crossing counts.

Segmental Level: This level represents the utterance as labeled acoustic segments. Although the set of labels may

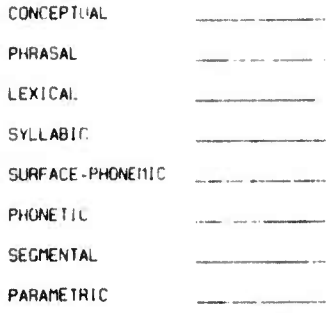


Fig. 1. Levels in HSII-C0.

be phonetic-like, the level is not intended to be phonetic—the segmentation and labeling reflect acoustic manifestation and do not, for example, attempt to compensate for the context of the segments or attempt to combine acoustically dissimilar segments into (phonetic) units.

As with all levels, any particular portion of the utterance may be represented by more than one competing hypothesis (i.e., multiple segmentations and labelings may co-exist).

Phonetic Level: At this level, the utterance is represented by a phonetic description. This is a *broad* phonetic description in that the size (duration) of the units is on the order of the "size" of phonemes; it is a *fine* phonetic description to the extent that each element is labeled with a fairly detailed allophonic classification (e.g., "stressed, nasalized [I]").

Surface-Phonemic Level: This level, named by seemingly contradicting terms, represents the utterance by phoneme-like units, with the addition of modifiers such as stress and boundary (word, morpheme, syllable) markings.

Syllabic Level: The unit of representation here is the syllable.

Lexical Level: The unit of information at this level is the word. (Note again that at any level competing representations can be accommodated.)

Phrasal Level: Syntactic elements appear at this level. In fact, since a level may contain arbitrarily many "sublevels" of elements structured as a modified AND/OR graph, traditional kinds of syntactic trees can be directly represented here.

Conceptual Level: The units at this level are "concepts." As with the phrasal level, it may be appropriate to use the graph structure of the data base to indicate relationships among different concepts.

The basic unit in the data structure is a *node*; a node represents the *hypothesis* that a particular element exists in the utterance. For example, a hypothesis at the phonetic level may be labeled as "I." Besides containing the hypothesis element name, a node holds several other kinds of information, including: 1) a correlation with a particular *time period* in the speech utterance,[5] 2) *scheduling parameters* (validity ratings, attention-focusing factors, measures of computing effort expended, etc.), and 3) *connection information* which relates the node to other nodes in the data base.

Structural relationships between nodes (hypotheses) are represented through the use of *links*; links provide a means for specifying contextual abstractions about the relationships of hypotheses. A link is an element in the data structure which associates two nodes as an ordered pair; one of the nodes is termed the *upper hypothesis*, and the other is called the *lower hypothesis*. The lower hypothesis is said to *support* the upper hypothesis; the upper hypothesis is called a *use* of the lower one. There are several types of links; in general, if a node serves as the upper hypothesis for more than one link, all of those links must be of the same type. Thus, one can talk of the "type of the hypothesis," which is the same as the type of all of its lower links. The two most important structural relationship types are SEQUENCE and OPTION.

A SEQUENCE node is a hypothesis that is supported by a (timewise) sequential set of hypotheses at a lower level (or sublevel—see below). For example, Fig. 2(a) shows a hypothesis of "will" at the lexical level supported by the (time-)ordered contiguous sequence of "W," "IH," and "L" at the surface-phonemic level. The interpretation of a SEQUENCE node is that all of the lower hypotheses must be valid in order to support the upper hypothesis.

An OPTION node is a hypothesis that has alternative supports from two or more hypotheses, each of which covers essentially the same time period. For example, Fig. 2(b) shows the hypothesis of "noun" at the phrasal level as being supported by any of "boy," "toy," or "tie," all of which are competing word hypotheses covering (approximately) the same time area.[6]

Fig. 3 is a example of a larger fragment of the global data structure. The level of a hypothesis is indicated by its vertical position, the names of the levels are given on the left. Time location is approximately indicated by horizontal placement, but duration is only very roughly indicated (e.g., the boxes surrounding the two hypotheses at the phrasal level should be much wider). Alternatives are indicated by proximity; for example, "will" and "would" are word hypotheses covering the same time span. Likewise, "question" and "modal-question," "you1" and "you2," and "J" and "Y" all represent pairs of alternatives.

This example illustrates several features of the data structure.

The hypothesis "you," at the lexical level, has two alternative phonemic "spellings" indicated; the hypothe-

[5] This time period is specified with an explicit estimation of fuzziness, even to the extent of spanning the entire utterance. As an extreme example, if a sequence of "phonemes" is being hypothesized from a "word" at the lexical level, the time boundaries of the phonemes are specified as "fuzzy" if information regarding their actual locations is not yet available.

[6] In addition to SEQUENCE and OPTION, there are two kinds of structural relationships which are generalizations of them: An AND node is similar to a SEQUENCE node except that there is no implication of any time sequentiality among the supports—they may overlap or be disjoint. An OR node is similar to an OPTION node in that the supports represent alternatives, but (as with the AND node) there is no time requirement.

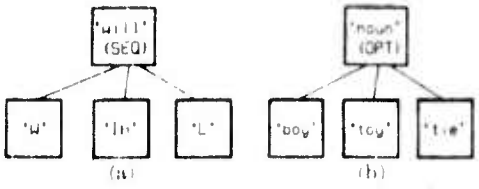


Fig. 2. Examples of SEQUENCE and OPTION relationships.

ses labeled "you1" and "you2" are nodes created, also at the lexical level, to hold those alternatives. In general, such *sublevels* may be created arbitrarily.

The link between "you1" and "D" is a special kind of SEQUENCE link (indicated here by a dashed line) called a CONTEXT link; a CONTEXT link indicates that the lower hypothesis supports the upper one and is contiguous to its brother links, but it is not "part of" the upper hypothesis in the sense that it is not within the time interval of the upper hypothesis—rather, it supplies a context for its brother(s). In this case, one may "read" the structure as stating "'you1" is composed of "J" followed by "AX" (schwa) in the context of the preceding "D."" (This reflects the phonological rule that "would you" is often spoken as "would-ja.") Thus, a CONTEXT link allows important contextual relationships to be represented without violating the implicit time assumptions about SEQUENCE nodes.

Whereas the phonemic spelling of the word "you" held by "you1" includes a contextual constraint, the "you2" option does not have this constraint. However, "you1" and "you2" are such similar hypotheses that there is strong reason for wanting to retain them as alternative options under "you" (as indicated in Fig. 3), rather than representing them unconnectedly. In general, the problem is that the use of a hypothesis implies certain contextual assumptions about its environment, while the support of a hypothesis may itself be predicated on a particular set of contextual assumptions. A mechanism, called a *connection matrix*, exists in the data structure to represent this kind of relationship by specifying, for an OPTION hypothesis, which of its alternative supports are applicable ("connected") to which of its uses. In this example, the connection matrix of "you" (symbolized in Fig. 3 by the two-dimensional binary matrix in the node) specifies that support "you1" is relevant to use "question" (but not to "modal-question") and that support "you2" is relevant to both uses. The use of a connection matrix allows the efforts of preceding KS decisions to be accumulated for future use and modification without necessitating contextual duplication of parts of the data base. Thus, much of the duplication of effort due to the backtracking mode of HSI is avoided in HSII.

Besides showing structural relationships (i.e., that one unit is composed of several other units), a link is a statement about the degree to which one hypothesis implies (i.e., gives evidence for the existence of) another hypothesis. The strength of the implication is held as information on the link. The sense of the implication may be negative; that is, a link may indicate that one hypothesis is evidence

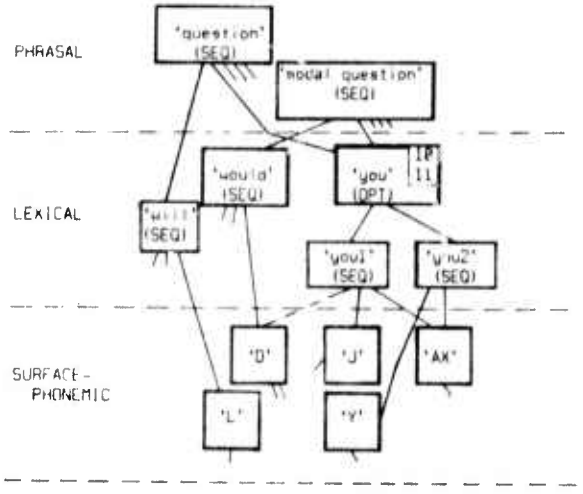


Fig. 3. Example of the data structure.

for the invalidity of another. Finally, this statement of implication is bidirectional; the existence of the upper hypothesis may give credence to the existence of the lower hypothesis and vice versa.

The nature of the implications represented by the links provides a uniform basis for propagating changes made in one part of the data structure to other relevant parts without necessarily requiring the intervention of particular KS's at each step. Considering the example of Fig. 3, assume that the validity of the hypothesis labeled "J" is modified by some KS (presumably operating at the phonetic level) and becomes very low. One possible scenario for rippling this change through the data base is the following.

First, the estimated validity of "you1" is reduced, because "J" is a lower hypothesis of "you1."

This, in turn, may cause the rating of "you" to be reduced.

The connection matrix at "you" specifies that "you1" is not relevant to "modal-question," so the latter hypothesis is not affected by the change in rating of the former. Notice that the existence of the connection matrix allows this decision to be made locally in the data structure, without having to search back down to the "D" and "J."

"Question," however, is supported by "you1" (through the connection matrix at "you"), so its rating is affected.

Further propagations can continue to occur, perhaps down the other SEQUENCE links under "question" and "you1."

KS Specification: A knowledge source is specified in three parts: 1) the conditions under which it is to be activated (in terms of the data base conditions in which it is interested, as described in the section on "preconditions" below), 2) the kinds of changes it makes to the global data base, and 3) a procedural statement (program) of the algorithm which accomplishes those changes. A KS is thus defined as possessing some processing capability which is able to solve some subproblem, given appropriate circumstances for its activation.

The decomposition of the overall recognition task into

various knowledge sources is regarded as being a natural decomposition. That is, the units of the decomposition represent those pieces of knowledge which can be distinguished and recognized as being somehow naturally independent.[7] Given a sufficient set of such KS's (that is, a set that provides enough overall connectivity among the various levels of the data base that a final recognition can be attained), the collection is called the "overall recognition system." Such a scheme of "inverse decomposition" (or, composition) seems very natural for many problem-solving tasks, and it fits well into the hypothesize-and-test approach to problem-solving. As long as a sufficient "covering set" for the data base connections is maintained, one can freely add new KS's, or replace or delete old ones. Each KS is in some sense self-contained, but each is expected to cooperate with the other KS's that happen to be present in the system at that time.

As examples of KS's, Fig. 4 shows the first set of processes implemented for HSH-C0. The levels are indicated as horizontal lines in the figure and are labeled at the left. The KS's are indicated by arcs connecting levels; the starting point(s) of an arc indicates the level(s) of major "input" for the KS, and the end point indicates the "output" level where the KS's major actions occur. In general, the action of most of these particular KS's is to create links between hypotheses on its input level(s) and: 1) existing hypotheses on its output level, if appropriate ones are already there, or 2) hypotheses that it creates on its output level.

The *Segmenter-Classifier* KS uses the description of the speech signal to produce a labeled acoustic segmentation. (See [7] for a description of the algorithm being used.) For any portion of the utterance, several possible alternative segmentations and labels may be produced.

The *Phone Synthesizer* uses labeled acoustic segments to generate elements at the phonetic level. This procedure is sometimes a fairly direct renaming of an hypothesis at the segmental level, perhaps using the context of adjacent segments. In other cases, phone synthesis requires the combining of several segments (e.g., the generation of [t] from a segment of silence followed by a segment of aspiration) or the insertion of phones not indicated directly by the segmentation (e.g., hypothesizing the existence of an [l] if a vowel seems velarized and there is no [l] in the neighborhood). This KS is triggered whenever a new hypothesis is created at the segmental level.

The *Word Candidate Generator* uses phonetic information (primarily just at stressed locations and other areas of high phonetic reliability) to generate word hypotheses. This is accomplished in a two-stage process, with a stop at the syllabic level, from which lexical retrieval is more effective.

[7] The approach taken in knowledge source decomposition is not an attempt to characterize somehow the overall recognition process and then apply some sort of traffic flow analysis to its internal workings in order to decompose the total process into minimally interacting KS's. Rather, KS's are defined by starting with some intuitive notion about the various pieces of knowledge which could be incorporated in a useful way to help achieve a solution.

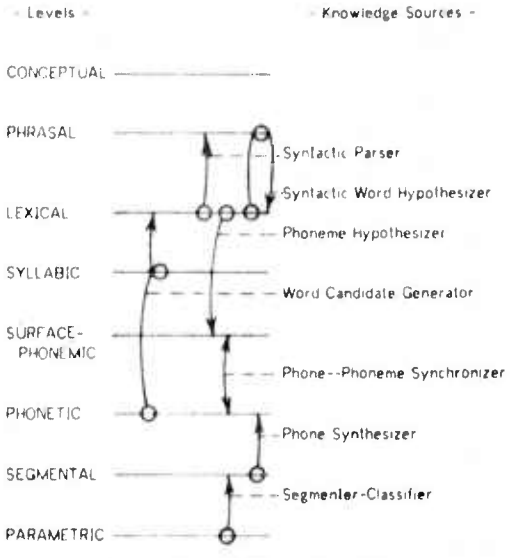


Fig. 4. First KS's for HSH-C0.

The *Syntactic Word Hypothesizer* uses knowledge at the phrasal level to predict possible new words at the lexical level which are adjacent (left or right) to words previously generated at the lexical level. ([19] contains a description of the probabilistic syntax method being used here.) This KS is activated at the beginning of an utterance recognition attempt and, subsequently, whenever a new word is created at the lexical level.

The *Phoneme Hypothesizer* KS is activated whenever a word hypothesis is created (at the lexical level) which is not yet supported by hypotheses at the surface-phonemic level. Its action is to create one or more sequences at the surface-phonemic level which represent alternative pronunciations of the word. (These pronunciations are currently prespecified as entries in a dictionary.)

The *Phone–Phoneme Synchronizer* is triggered whenever a hypothesis is created at either the phonetic or the surface-phonemic level. This KS attempts to link up the new hypothesis with hypotheses at the other level. This linking may be many-to-one in either direction.

The *Syntactic Parser* uses a syntactic definition of the input language to determine if a complete sentence may be assembled from words at the lexical level.

Fig. 5 shows the initial HSH-C0 KS's of Fig. 4 augmented with four additional ones which are being implemented or are planned.

The *Semantic Word Hypothesizer* uses semantic and pragmatic information about the task (news retrieval, in this case) to predict words at the lexical level.

The *Phonological Rule Applier* rewrites sequences at the surface-phonemic level. This KS is used: 1) to augment the dictionary lookup of the Phoneme Hypothesizer, and 2) to handle word boundary conditions that can be predicted by rule.

The primary duties of the *Segment-Phone Synchronizer* and the *Parameter-Segment Synchronizer* are similar: to recover from mistakes make by the (bottom-up) actions of the Phone Synthesizer and Segmenter–Classifier, re-

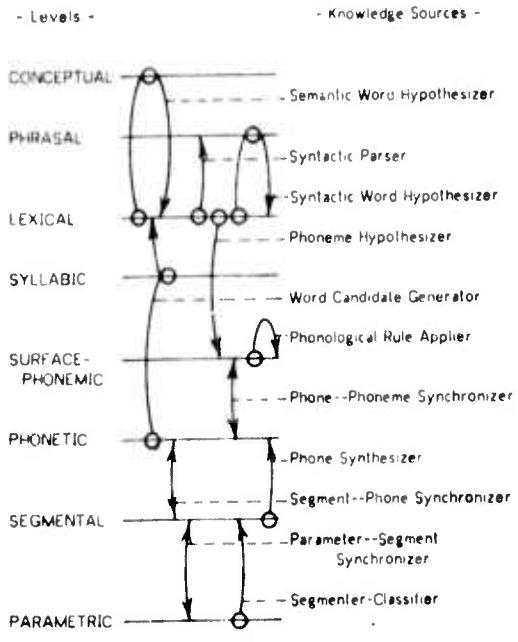


Fig. 5. Augmented KS set for HSII-C0.

spectively, by allowing feedback from the higher to the lower level.

The introduction of these knowledge sources indicates the modularity and extendability of the system in terms of both KS's and levels. In particular, notice that even though the purpose of some KS is stated as "correcting errors produced by other KS's," each KS is independent of the others. Yet additional KS's will be added to the configuration as the need for them is seen and as ideas for their implementation are developed.

In addition to the KS modules described above, all of which embody speech knowledge, several policy modules exist. These modules, which interface to the system in a manner identical to the speech modules, execute policy decisions, e.g., propagation of ratings, focusing of attention, and scheduling of other modules.

Matching-Prototypes and Associative Retrieval: The synchronous processing activity in HSII is primarily data-directed; this implies the need for some mechanism whereby one can retrieve parts of the global data base in an associative manner. HSII provides primitives for associatively searching the data base for hypotheses satisfying specified conditions (e.g., finding all hypotheses at the phonetic level which contain a vowel within a certain time range). The search condition is specified by a *matching prototype*, which is a partial specification of the components of a hypothesis. This partial specification permits a component to be characterized by: 1) a set of desired values, or 2) a DON'T-CARE condition, or 3) values of components of a hypothesis previously derived by matching another prototype. A matching prototype can be compared against a set of hypotheses. Those hypotheses whose component values match those specified by the matching prototype are returned as the result of the search. Associative retrieval of structural relationships among hypotheses is also provided by several primitives. More complex retrievals can be accomplished by combining the retrieval primitives in appropriate ways.

Preconditions and Change Sets: Associated with every KS is a specification of the data base conditions required for the instantiation of that KS. Such specifications, called *preconditions*, conceptually form an AND/OR tree composed of matching prototypes and structural relationships which, when applied to the data base in an associative manner, detect the regions of the data base in which the KS is interested (if the precondition is capable of being satisfied at that time). Alternatively, one might think of the precondition specification as a procedure, involving matching prototypes and structural relationships, which effectively evaluates a conceptual AND OR tree. This procedure may contain arbitrarily complex decisions (based on current and past modifications to the data base) resulting in the activation of desired KS's within the chosen contexts. The context corresponding to the discovered data base region which satisfies some KS's precondition is used as an initial context in which to instantiate that KS as a new process. If there are multiple regions in the data base that satisfy the specified conditions, the KS can be separately instantiated for each context, or once with a list of all such contexts.

Whenever any KS performs a modification to the global data base, the essence of the modification is preserved in a *change set* appropriate to the change made (e.g., one change set records rating changes, while another records time range changes). Change sets serve to categorize data base modifications (events) and are thus useful in precondition evaluation since they limit the areas in the data base that need to be examined in detail. As currently implemented, precondition evaluators exist as a set of procedures which monitor changes in the data base (i.e., they monitor additions to those change sets in which they are interested). These precondition procedures are themselves data-directed in that they are applied whenever sufficient changes have been made in the global data base. In effect, the precondition procedures themselves have preconditions, albeit of a much simpler form than those possible for KS's. For example, a precondition procedure may specify that it should be invoked (by a system precondition invoker) whenever changes occur to two adjacent hypotheses at the word level or whenever support is added to the phrasal level. By using the (coarse) classifications afforded by change sets, the system avoids most unnecessary data base examinations by the precondition procedures. The major point to be made is that the scheme of precondition evaluation is *event driven*, being based on the occurrence of changes in the global data base. In particular, precondition evaluators are not involved in a form of busy waiting in which they are constantly looking for something that is not yet there.

Once invoked, a precondition procedure uses sequences of associative retrievals and structural matches on the data base in an attempt to establish a context satisfying the preconditions of one or more of "its" KS's; any given

precondition procedure may be responsible for instantiating several (usually related) KS's. Notice that the data-directed nature of precondition evaluation and KS instantiation is linked closely to the primitive functions that are able to modify the data base, for it is only at points of modification that a precondition that was unsatisfied before may become satisfied. Hence, data base modification routines have the responsibility (although perhaps indirectly) of activating the precondition evaluation mechanism.[8]

Multiprocessing Considerations: Some Problems and Their Solutions

A primary goal in the design of HSII is to exploit the potential parallelism of the Hearsay system model as fully as possible. Several issues associated with the introduction of parallel KS processes into HSII will be described and their current solutions outlined.

Local Contexts: A precondition evaluator (process) is invoked based on the occurrence of certain changes which have taken place in the global data base since the last time the evaluator was invoked; these changes, together with the state of the relevant parts of the global data base at the instant at which the precondition evaluator is invoked, form a *local context* within which the evaluator operates. Conceptually, at the instant of its invocation, the precondition evaluator takes a snapshot of the global data base and saves the substance of the changes that have occurred to that moment, thereby preserving its local context.

The necessity of preserving this local context exists for several reasons: 1) HSII consists of asynchronous processes sharing a common global data base which contains only the most current data (that is, no history of data modification is preserved in the global data base), 2) since the precondition evaluators are also executed asynchronously, each evaluator is interested only in changes in the data base which have occurred since the last time that particular evaluator was executed (that is, the relevant set of changes for a particular precondition evaluator is time-dependent on the last execution of that evaluator), and 3) further modifications to the global data base which are of interest to a given KS process may occur between the time of that KS process's instantiation and the time of its completion (in particular, such modifications and their relationship to data base values which existed at the time of the instantiation of the KS process may influence the computation of that KS process). Hence, the problem of creation of local contexts exists, as do the associated problems of signaling a KS process

[8] One might think of HSII as a production system [14] which is executed asynchronously. The preconditions correspond to the left-hand sides (conditions) of productions, and the knowledge sources correspond to the right-hand sides (actions) of the productions. Conceptually, these left-hand sides are evaluated continuously. When a precondition is satisfied, an instantiation of the corresponding right-hand side of its production is created; this instantiation is executed at some arbitrary subsequent time (perhaps subject to instantiation scheduling constraints).

T1: start precondition evaluator PRE
(triggered by data base changes)
T2: PRE instantiates a knowledge-source process KS
T3: end PRE

T4: start KS
T5: after KS revalidation of precondition
<computation>
T6: KS modifies global data base
<computation>
T7: KS modifies global data base again
T8: end KS

Fig. 6. Timing sequence of a precondition evaluator and KS.

when its local context is no longer valid and of updating these contexts as further changes occur in the global data base.

Consider the time sequence of events shown in Fig. 6. The precondition evaluator PRE is activated to respond to changes occurring in the global data base. PRE should execute in the context of changes existing at time T1 (since that context contains the changes which caused PRE to be activated). KS is instantiated (readied for running) at T2 due to further conditions PRE discovered about the change context of T1. Hence, PRE should pass the context of T1 as the initial environment in which to run KS.

By time T4, when KS actually starts to execute, other changes could have occurred in the global data base due to the actions of other KS processes. So KS should examine these new updating changes (those occurring between T1 and T4) and revalidate its precondition, if necessary. After revalidation, KS assumes the updated context of T5, and it proceeds to base its computation on the context of changes as of T5.

When KS wishes to perform an actual update of elements of the global data base at T6, it must examine the changes to the global data base that have occurred between T5 and T6 to see if any other KS processes may have violated KS's preconditions, thereby invalidating its computations. Having performed this revalidation and any data base updating, KS should update its context to reflect changes up to T6 for use in its further computation. At T7, KS must look for further possible invalidations to its most recent computations, due to possible changes in the global data base by other KS processes during the time period T6 to T7. When KS (which is an instantiation of some KS) completes its actions at T8, its local context may be deleted.

Changes occurring between instantiations of a KS are accumulated in the local contexts of the precondition evaluators and may become part of the local context of a future instantiation of a KS. Thus, the precondition evaluators are always collecting data base changes (since these evaluators are permanently instantiated), while individual KS instantiations accumulate data base changes only during their transient existence.

In practice, to create local contexts one need only save the contents of *changes* which occur in the global data base (thereby allowing the global data base to contain only the very latest values). In particular, no massive copy operations involving the global data base are required.

Thus, for each data base event caused by a modification primitive, the associated change[9] is distributed (copied) into the local contexts (which can now be characterized as *local change sets*,[10] referring to the previous discussion on change sets) of all KS processes and precondition evaluators who care. Notice that not every KS process and precondition evaluator cares about every change that takes place. For example, a fricative detector will not care about a data base change associated with the grouping of several words to form a syntactic phrase, but it is interested in the hypothesization of a word whose phonemic spelling contains a fricative.

In order for a KS (or precondition evaluator) process to be notified of changes which occur to particular fields of particular nodes which are in its local context, those fields need to be *tagged* with an alias (called a *tagid*) belonging to that KS instantiation. Then, whenever a modification is made to the global data base, a *message* signaling the change is sent to all who have tagged the field being changed. In addition, the contents of the change are also distributed to the local contexts of those KS's receiving the message. This data field tagging may be requested by either a precondition evaluator which is about to instantiate a KS based on the values of particular fields (which represent the context satisfying the precondition), or by a KS process, once instantiated, which may request additional fields to be tagged.

For example, in its search through the global data base for conditions satisying the preconditions of some KS, a precondition evaluator may accumulate references to the data fields which it has examined; and when the entire precondition has been found to be satisfied, the precondition evaluator tags those fields (for which it has accumulated references) with the name of the KS process it is about to instantiate. Similarly, having been invoked, a KS process might wish to do certain computations, but only if certain data fields are not altered; the KS process itself can tag these fields and thereby be informed of subsequent tampering with the tagged fields. Subsets of these tagged fields (the subsets being formed according to the tagid's chosen) form *critical sets* (specifying the fields of the local context for the KS process) which are *locked* (see below) every time the KS process wishes to modify the global data base. Thus, after having locked the critical set and prior to performing any update operations, the KS process can check to see whether any other KS process has made any changes which might invalidate the current KS's assumed context (and hence, perhaps, invalidate its proposed update).[11] For example, if a KS process is verifying a hypothesis in the data base because its rating exceeds 50 (i.e., the rating value represents part of the local context for the KS process), then before performing any modifications on the data base which depend on the assumption, the KS process should check to be sure that no other KS process has invalidated the rating assumption in the meantime.

In addition to maintaining local copies of the various relevant changes that have occurred since the instantiation of a KS process, a KS process may also include in its context various *supersets* of these changes. For example, if a knowledge source is interested in the values of changes in a node's begin-time or end-time, it might be worthwhile to include a superset which accumulates indicators pointing to changes in either begin-time or end-time (since checking such a superset for new elements would be quicker than checking each constituent subset individually). Note that supersets do not themselves contain any data values, but rather are pointers to changes contained in some one of its subsets. Further note that a KS need not contain in its local context all (or any) of the constituent subsets of a superset in order to have that superset in its local context.

Data Base Deadlock Prevention: Any KS process may request exclusive access to some collection of fields at any time. Thus, unless some care is taken in ordering the requests for such exclusive access, the possibility of getting into a deadlock situation exists (where, for example, one KS process is waiting for exclusive access to a field already held exclusively by a second KS process, and vice versa, resulting in neither process being able to proceed). Applying a linear ordering to the set of lockable fields and requesting exclusive access according to that ordering is a commonly used means of deadlock avoidance in resource allocation. However, this technique works only if all the resources (fields) to be locked are known ahead of time. The ability to tag a data field allows a KS process to locate and examine in arbitrary order the set of hypotheses that will form the context for a data base modification and then, only after the entire set of desired hypotheses (and links) has been determined, to lock the desired set and perform the modification.

Regarding the global data base as having orthogonal dimensions of information level versus utterance time further allows a convenient means of locking entire regions of the data base without explicitly naming the nodes or fields to be affected. This "blanket-locking" is called *region locking*. For example, a KS process might request the locking of a region consisting of the time interval from 50 to 80 centiseconds at the phonetic level. To assist in de-

[9] The information which defines the change consists of the locus of the change (i.e., a node name and a field name) and the old value of the field.

[10] An alternative to replicating the change information for the various KS processes is to maintain a single central copy of those data, passing only references to the centralized items to the various local contexts. The individual change items may then be deleted from their central change sets whenever there are no further outstanding references to that change.

[11] The characterization of local contexts according to specific data fields (which is made possible in part by the choice of levels in the global data base) helps to minimize the overhead associated with context maintenance. Also, the smaller the context, the more flexible the scheduling strategy may be (since it needs to be less concerned with the time required for a context swap on a processor).

termining such regions, each node in the global data base contains a bit vector whose high-order bits describe the time regions covered by the node and whose low-order bits contain the integer level number of that node. These region fields may be manipulated to allow locking of various regions in the data base. Note that a region-lock may be used to obtain exclusive access to a time region which need not contain any nodes yet (i.e., one may region-lock a "vacuum").

To eliminate the possibilities of deadlock, the actual locking operation is relegated to a system primitive, and a KS process is required to present to the locking primitive the entire set of fields to which it wants exclusive access. This set is then extended to include all fields in the critical set of the calling KS process, for the reasons relating to context revalidation given above. The system locking primitive can then request exclusive access for this union of data fields, on behalf of the KS process, according to a universal ordering scheme (such an ordering being possible since every node in the global data base essentially has a unique serial number) which assures that no deadlocks occur. Having been granted exclusive access to all desired fields at once, the KS process may then check to see whether there have been any changes to the tagged data fields. If there have been none, it can proceed to perform its modifications (which modifications are sent to the local contexts of others who care about such things). However, if there were changes, the KS process can, after examining the changed data fields, decide whether it still wants to perform the modifications. When the KS process is finished updating the data base, it releases all its locked fields by executing a system primitive unlocking operator. In particular, the system ensures that a KS process will not request two lock operations without issuing an intervening unlock operation. In this manner, any possibility of a data base deadlock is eliminated.

Goal-Directed Scheduling: The computational sequence of a typical HSII run may be characterized by considering the chronological sequence of states of the utterance recognition at any particular data base level. For example, if one considers the efforts in producing the word level and traces the development of the "best" partial sentences, the processing that will have been done is analogous to a general tree-search, where each node of the tree represents some partially completed sentence (with the eventual resultant sentence being one of the leaves of the tree). The problem now is to guide this tree searching so as to find the answer leaf in an optimal way (according to some measure of optimality). To achieve this end, *ratings* are associated with every hypothesis and link in the global data base (and thus with every partial sentence node of the analogous search tree). Using these ratings (which are effectively evaluation functions indicating the goodness of the work done so far with respect to a given partial sentence), one may employ tree-searching strategies to advance the search in some optimal manner.

To complicate matters further, however, HSII is intended to be a multiprocess-oriented system. Therefore, schemes must be devised for effectively searching a problem-solving graph using parallel processes, since one can conceive of pursuing several branches of the search graph in parallel by asynchronously instantiating various KS processes to evaluate various alternative paths. One must also take into consideration the underlying hardware architecture. The physical placement of the global data base and the KS processes will have a very definite influence on the scheduling philosophy chosen and the resultant system efficiency.

To aid in making scheduling decisions, one may associate with every node in the global data base some *attention-focusing factors*, which are indicators telling how much effort has been devoted to processing in this area of the search tree and how desirable it is to devote further effort to this section of the tree. Such attention-focusing factors may also be associated with various speech time regions to indicate interest in doing further processing on certain regions of the utterance, regardless of any particular information level. Furthermore, attention-focusing factors are associated with every data base modification, thereby distributing attention-focusing factors to the various change sets which constitute the local contexts of the processes in the system. The scheduler is one such process which might be especially interested in such focusing factors, as will be described below. The use of the various ratings and attention-focusing factors allows HSII to perform *goal-directed scheduling*, which is process scheduling so as to achieve "optimal" recognition efficiency. The thrust of goal-directed scheduling is that, while there may be many processes ready to run and work on various parts of the search tree, one should first schedule those processes which can best help to achieve the goal of utterance recognition. Notice that such search-tree pruning techniques as the alpha–beta procedure (which is essentially a sequential algorithm anyway) do not apply to HSII's nongame search trees, which do not have the constraint that alternating levels of the tree represent the moves of an opponent.

Goal-directed scheduling may be viewed as having two separate functions: 1) using the ratings and attention-focusing factors associated with the global data base components to schedule *KS processes* which have been invoked (readied for execution) in response to events previously detected in the global data base, and 2) using these same attention-focusing factors to detect important areas in the global data base which require further work, and invoking *precondition evaluators* as soon as possible to instantiate new KS processes to work in those important areas. Thus, the attention-focusing factors within the global data base serve to schedule both KS processes and precondition evaluators.

A KS process might be scheduled for execution because it possesses the only processing capability available to be

applied to an important unexplored area of the data base. However, if there are many such processes ready to execute, the scheduler can perform a type of *means-ends* analysis in which those KS processes are scheduled which are likely to produce data base changes in which the system is currently interested (such interest being noted by high attention-focusing factors in a given change set). For example, if the data base contains focusing factors which highlight activity in a time region in which there are no structural connections between two adjoining levels, the scheduler would probably give a higher priority to a KS process which will attempt (as indicated in its external specifications) to make such a connection than to a KS process which is likely merely to perform a minor refinement on the ratings in one of the levels.

Another means of effecting goal-directed scheduling relates to the attention-focusing factors associated with various time regions of the utterance (such focusing factors reaching across all the information levels of the global data base). Using these time-region focusing factors, one can schedule KS processes which can contribute in a particular time region, or invoke precondition evaluators to instantiate some new KS processes to work within the desired time region.

SUMMARY, CURRENT STATUS, AND PLANS

In summary, HSII is a system organization for speech understanding that permits the representation of speech knowledge in terms of a large number of diverse KS's which cooperate via a generalized (in terms of both data and control) form of the hypothesize-and-test paradigm. KS's are independent and separable; they are activated in a data-directed manner and execute asynchronously, communicating information among themselves through a global data base. This global data base, which is a representation of the partial analysis of the utterance, is a three-dimensional data structure (in which the dimensions are level of representation, time, and alternatives) augmented by AND/OR structural relationships which interconnect elements of the data structure. This global data base structure permits information generated by one KS to be: 1) retained for use by other KS's, and 2) quickly propagated to other relevant parts of the data base. In addition to being a new representational framework for specifying speech knowledge, HSII is a system organization suitable for efficient implementation on a multiprocessor computer system. In particular, the system organization employs techniques which, while not violating the independence and modularity of KS's, permit: 1) avoidance of deadlock in the data base, 2) efficient implementation of data-directed sequencing of knowledge sources, and 3) goal-directed scheduling of asynchronously executable KS processes.

A preliminary, synchronous version of HSII has been operating on Carnegie-Mellon University's PDP-10 since January 1974. The fully asynchronous, multiprocess version of HSII is now in the final stages of being implemented, also on the PDP-10, and is expected to be running by August 1974. This multiprocess version of HSII will also contain the capability of simulating the effects of operating HSII in a multiprocessor environment. Experience with this multiprocess version of HSII, together with simulation data on the effects of operating in a multiprocessor environment, will form the basis for a multiprocessor version of HSII on a multi-miniprocessor computer (C.mmp). An initial implementation of HSII on C.mmp is expected to be completed in the first quarter of 1975.

A Carnegie-Mellon University technical report providing detailed examples of HSII's recognition process is planned for Autumn 1974.

APPENDIX I

HSI PERFORMANCE SUMMARY

Fig. 7 summarizes the performance of the HSI system as of Autumn 1973. The results are based on tests of 144 sentences containing 676 words, spoken by five speakers and using four different tasks (Chess, News Retrieval, Medical Questionnaire, and Desk Calculator) whose vocabularies range from 28 to 76 words. (More complete descriptions of the tasks are given in [5] and [12].)

Column 1 gives the data set number, task, and speaker identification. All of the speakers are male adults. Data set 4 was recorded over a long-distance telephone connection; the others were recorded in a fairly quiet environment using one of several medium to high-quality microphones. All of the utterances represent speech read from a script (as opposed to being spontaneous).

Column 2 gives the number of words in the task lexicon.

Column 3 shows the number of sentences in the data set; column 4 gives the total number of word tokens in the set.

Column 5 contains the results of recognition with just the Acoustics module (the acoustic-phonetic source of knowledge). The task lexicon is the sole restriction—the Syntax and Semantics modules are not used. The first subcolumn indicates the percent of times the correct word appears as the first choice of the Acoustics module in the hypothesis list. The second subcolumn indicates the percent of times the correct word is in the top two choices; the third shows the top three. In this mode each incorrect word recognition is overridden by specifying the correct word boundary positions (using carefully hand-segmented data) so that errors do not propagate.

Column 6 gives the results for the HSI system recognition with the Acoustics module and the Syntax module both operating. The first subcolumn indicates the percent of sentences recognized completely correctly. The "near-miss" (indicated below that number in the first subcolumn) indicates the percent of times the recognized utterance differed from the actual utterance by at most one word of

1	2	3	4	5 ACOUSTIC MODE			6 ACOUSTIC AND SYNTAX MODE				7 ACOUSTIC, SYNTAX, AND SEMANTIC MODE			
DATA SET: TASK / SPEAKER	WORDS IN LEX	#SEN	#WORD	%IN #1	%IN #1 +#2	%IN #1 +#2 +#3	/SEN %NERR	%WOR %SER	AVE T/SEN (Sec)	AVE T/S INP	/SEN %NERR	%WER %SER	AVE T/SEN	AVE T/S INP
1 CHESS/RM	31	14	82	61	79	85	43 100	87	11	6	100 100	100	5	5
2 CHESS/JB	31	19	86	44	69	78	74 95	93	12	9	100 100	100	8	6
3 CHESS/JB	31	21	105	38	48	73	15 58	69	15	8	46 90	88	13	7
4 CHESS/TEL/ BL	31	25	99	43	64	76	52 84	78	7	6	80 88	88	7	6
SUB-TOTALS		79	352	45	69	77	46 80	61	11	7	79 93	93	9	6
5 NEWS-RETREV/VL	28	19	104	30	53	67	71 87	63	13	7	xxx xxx	xxx	xxx	xxx
6 NEO-QUES/JR	76	21	92	20	38	49	74 33	40	13	11	xxx xxx	xxx	xxx	xxx
7 DESI-CALC/JB	38	25	128	42	63	72	35 55	68	28	14	xxx xxx	xxx	xxx	xxx
TOTALS		144	676	39	61	71	38 64	70	15	9	79 93	93	9	6

Fig. 7. Performance results of the HS1 system—Autumn 1973.

approximately similar phonetic structure. The second subcolumn gives the percent of words recognized correctly. The mean computation times on the PDP-10 computer (in seconds per sentence and in seconds per second of speech) are shown in subcolumns three and four.

Column 7 shows results for recognition using all three sources of knowledge (for the Chess task only): the Acoustics, Syntax, and Semantics modules. The subcolumns are similar to those of Column 6.

As one might expect, accuracy increases with the introduction of more KS's. It is also true that computation time decreases with more KS's; i.e., the additional KS's serve to reduce the size of the space that must be searched.

Performance differences across the different data sets are attributable to the following several sources:

1) Task lexicon size.

2) Phonetic content of the lexicons: many rules in the Acoustics module handle particular kinds of phonetic juxtaposition. Vocabularies as small as the ones used here do not exhaust all of these cases; each such vocabulary exhibits some cases not found in each of the others. Because a larger amount of effort has gone into the rules based on the Chess lexicon, that task also performs best.

3) Speaker differences.

It is encouraging that the performance using the telephone input (data set 4) does not appear to differ significantly from the other data sets, which use higher quality input. The labeling of acoustic segments is defined by a table of values derived from training syllables uttered by the speaker over the same channel used to input the test data; thus it is somewhat self-normalizing. Also, the system detects that there is very little higher frequency energy (in the case of the telephone input); this automatically triggers programmed readjustment of some thresholds and heuristics dealing with high frequency information.

ACKNOWLEDGMENT

We wish to acknowledge the contributions of the following people: A. Newell for his insights into the task of problem-solving, D. Parnas and L. Coopwider for their discussions on system design and decomposition, G. Gill for his untiring efforts in system implementation, and D. McCracken for his valuable criticism of the ideas as presented in this paper.

REFERENCES

[1] J. Baker, "The DRAGON system—an overview," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 22–26; also, this issue, pp. 24–29.

[2] J. Barnett, "A vocal data management system," *IEEE Trans. Audio Electroacoust. (Special Issue on 1972 Conference on Speech Communication and Processing)*, vol. AU-21, pp. 185–188, June 1973.

[3] C. G. Bell *et al.*, "C.mmp: the CMU multi-miniprocessor computer," Dep. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, Pa., Tech. Rep., 1971.

[4] C. G. Bell, R. C. Chen, S. H. Fuller, J. Grason, S. Rege, and D. P. Siewiorek, "The architecture and application of computer modules: a set of components for digital systems design," presented at COMPCON 7th, San Francisco, Calif., pp. 177–180, 1973.

[5] L. D. Erman, "An environment and system for machine understanding of connected speech," Ph.D. dissertation, Dep. Comput. Sci., Stanford Univ., Stanford, Calif., 1974.

[6] L. D. Erman, R. D. Fennell, V. R. Lesser, and D. R. Reddy, "System organizations for speech understanding: implications of network and multiprocessor computer architectures for AI," in *Proc. 3rd Int. Joint Conf. Artificial Intelligence*, Stanford, Calif., Aug. 1973, pp. 194–199.

[7] H. G. Goldberg, D. R. Reddy, and R. L. Suslick, "Parameter-independent machine segmentation and labeling," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 106–111.

[8] C. Hewitt, "Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot," Massachusetts Inst. Technol., Cambridge, project MAC, AI Memo. 251, 1972.

[9] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 255–260.

[10] M. J. Knudsen, "Real-time linear-predictive coding of speech on the SPS-41 microprogrammed triple-processor system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 274–277; also this issue, pp. 140–145.

[11] S. Kriz, "A 16-bit A-D-A conversion system for high-fidelity audio research," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 278–282; also, this issue, pp. 146–149.

[12] R. B. Neely, "On the use of syntax and semantics in a speech understanding system," Ph.D. dissertation, Stanford Univ., Stanford, Calif., 1973.

[13] A. Newell *et al.*, *Speech Understanding Systems: Final Report of a Study Group*, 1971. Amsterdam, The Netherlands: North-Holland, 1973.

[14] A. Newell, "Production systems: models of control structures," in *Visual Information Processing*, W. C. Chase, Ed. New York: Academic, 1973, pp. 463–526.

[15] D. R. Reddy, L. D. Erman, and R. B. Neely, "The C-MU speech recognition project," in *Proc. IEEE Conf. System Sciences and Cybernetics*, Pittsburgh, Pa., 1970.

[16] ——, "A model and a system for machine recognition of speech," *IEEE Trans. Audio Electroacoust. (Special Issue on 1972 Conference on Speech Communication and Processing)*, vol. AU-21, pp. 229–238, June 1973.

[17] D. R. Reddy, L. D. Erman, R. D. Fennell, and R. B. Neely, "The HEARSAY speech understanding system: an example of

the recognition processes," in *Proc. 3rd Int. Joint Conf. Artificial Intelligence*, Stanford, Calif., Aug. 1973, pp. 185–183.
[18] R. Reddy and A. Newell, "Knowledge and its representation in a speech understanding system," in *Knowledge and Cognition*, L. W. Gregg, Ed. Washington, D. C.: Erlbaum, ch. 10, to be published.
[19] E. Rich, "Inference and use of simple predictive grammars," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, p. 242.
[20] H. B. Ritea, "A voice-controlled data management system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 28–31.
[21] P. Rovner, B. Nash-Webber, and W. Woods, "Control concepts in a speech understanding system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 267–272; also this issue, pp. 136–140.
[22] J. F. Rulifson *et al.*, "QA4: a procedural calculus for intuitive reasoning," AI Center, Stanford Res. Inst., Menlo Park, Calif., Tech. Note 73, 1973.
[23] L. Shockey and L. D. Erman, "Sub-lexical levels in the Hearsay II speech understanding system," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, pp. 208–210.
[24] W. A. Woods, "Motivation and overview of BBN SPEECHLIS: an experimental prototype for speech understanding research," in *Proc. IEEE Symp. Speech Recognition*, Carnegie-Mellon Univ., Pittsburgh, Pa., Apr. 1974, p. 1–10; also this issue, pp. 2–10.

PARALLELISM IN AI PROBLEM SOLVING: A CASE STUDY OF HEARSAY II

R. D. Fennell and V. R. Lesser
Department of Computer Science[1]
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

October, 1975

ABSTRACT

The Hearsay II speech-understanding system (HSII) (Lesser, *et al.*, 1974; Fennell, 1975; Erman and Lesser, 1975) is an implementation of a knowledge-based multiprocessing AI problem-solving organization. HSII is intended to represent a problem-solving organization which is applicable for implementation in a multiprocessing environment, and is, in particular, currently being implemented on the C.mmp multiprocessor system (Bell, *et al.*, 1971) at Carnegie-Mellon University. The object of this paper is to explore several of the ramifications of such a problem-solving organization by examining the mechanisms and policies underlying HSII which are necessary for supporting its organization as a multiprocessing problem-solving system. First, an abstract description of a class of problem-solving systems is given using the Production System model of Newell (1973). Then, the HSII problem-solving organization is described in terms of this model. The various decisions made during the course of design necessitated the introduction of various multiprocessing mechanisms (e.g., mechanisms for maintaining data localization and data integrity), and these mechanisms are discussed. Finally, a simulation study is presented which details the effects of actually implementing such a problem-solving organization for use in a particular application area, that of speech understanding.

[1] This research was supported in part by the Defense Advanced Research Projects Agency of the Office of the Secretary of Defense (Contract F44620-73-C-0074) and monitored by the Air Force Office of Scientific Research.

INTRODUCTION

Many AI problem-solving tasks require large amounts of processing power in order to achieve solution in any given computer implementation of a problem-solving strategy. The amount of processing power required is directly related to the size of the search space which is examined during the course of problem solution. Exhaustive search of the state space associated with almost any problem of interest is precluded due to the sheer size of the state space.[1] In most problem-solving attempts, heuristics are employed which prune the search space to a more manageable size. However, searching even the reduced state space often requires large amounts of processing power. The demand for sufficient computing power becomes critical in tasks requiring real-time solution, as is the case in the speech-understanding task with which this paper is primarily concerned. For example, a speech-understanding system capable of reliably understanding connected speech involving a large vocabulary and spoken by multiple speakers is likely to require from 10 to 100 million instructions per second of computing power, if the recognition is to performed in real time.[2] Recent trends in technology suggest that this computing power can be economically obtained through a closely-coupled network of asynchronous "simple" processors (involving perhaps 10 to 100 of these processors), (Bell, *et al.*, 1973, and Heart, *et al.*, 1973). The major problem (from the problem-solving point of view) with this network multiprocessor approach for generating computing power is in devising the various problem-solving algorithms in such a way as to exhibit a structure appropriate for exploiting the parallelism available in the multiprocessor network, for it is only by taking advantage of this processing parallelism that the desired effective computing power will be achieved.

The Hearsay II speech-understanding system (HSII) (Lesser, *et al.* 1974; Fennell, 1975; and Erman and Lesser, 1975) currently under development at Carnegie-Mellon University represents a problem-solving organization that can effectively exploit a multiprocessor system. HSII has been designed as an AI system organization suitable for expressing *knowledge-based problem-solving strategies* in which appropriately

[1] As an example, consider the chess-playing task. In an end game situation, there are typically 20 legal moves at each ply (half-move); so for a search depth of 6 plies, the search space will have 64 million branches.

[2] The Hearsay I (Reddy, *et al.*, 1973a,b,c and Erman, 1974) and Dragon (Baker, 1975) speech understanding systems require approximately 10 to 20 mips of computing power for real-time recognition when handling small vocabularies.

organized subject-matter knowledge may be represented as *knowledge sources* capable of contributing their knowledge in a parallel data-directed fashion. A *knowledge source* may be described as an agent that embodies the knowledge of a particular aspect of a problem domain and is useful in solving a problem from that domain by performing actions based upon its knowledge so as to further the progress of the overall solution. It is felt that the knowledge source is an appropriate unit for use in the decomposition of a knowledge-intensive task domain. Knowledge sources, being suitably organized capsules of subject-matter knowledge, may be independently formulated as various pieces of the knowledge relevant to a task domain become crystallized. The HSII system organization allows these various independent and diverse sources of knowledge to be specified and their interactions coordinated so they might cooperate with one another (perhaps asynchronously and in parallel) to effect a problem solution. As an example of the decomposition of a task domain into knowledge sources, in the speech task domain there might be distinct knowledge sources to deal with acoustic, phonetic, lexical, syntactic, and semantic information. While the speech task is the first test of the multiprocessing problem-solving organization of HSII, it is believed that the system organization provided by HSII is capable of expressing other knowledge-based AI problem-solving strategies, as might be found in vision, robotics, chess, natural language understanding, and protocol analysis. In fact, proposals are under way which will further test the applicability of HSII by implementing a system for the analysis of natural scenes using the HSII problem-solving organization (Ohlander, 1975).

The rest of this paper will explore several of the ramifications of such a problem-solving organization by examining the mechanisms and policies underlying HSII which are necessary for supporting its organization as a multiprocessing problem-solving system. First, an abstract description of a class of problem-solving systems is given using the Production System model of Newell (1973). Then, the HSII problem-solving organization is described in terms of this model. The various decisions made during the course of design necessitated the introduction of various multiprocessing mechanisms (e.g., mechanisms for maintaining data localization and data integrity), and these mechanisms are discussed. Finally, a simulation study is presented which details the effects of actually implementing such a problem-solving organization in a multiprocessor environment.

THE MODEL

An Abstract Model for Problem Solving

In the abstract, the problem-solving organization underlying HSII may be modeled in terms of a "production system," (Newell, 1973). A *production system* is a scheme for specifying an information processing system in which the control structure of the system is defined by operations on a set of *productions* of the form 'P → A', which operate from and on a collection of data structures. 'P' represents a logical antecedent, called a *precondition*, which may or may not be satisfied by the information encoded within the dynamically current set of data structures. If 'P' is found to be satisfied by some data structure, then the associated *action* 'A' may be executed, which presumably will have some altering effect upon the data base such that some other (or the same) precondition becomes satisfied. This paradigm for sequencing of the actions can be thought of as a data-directed control structure, since the satisfaction of the precondition is dependent upon the dynamic state of the data structure. Productions are executed as long as their antecedent preconditions are satisfied, and the process halts either when no precondition is found to be satisfied or when an action executes a stop operation (thereby signalling problem solution or failure, in the case of problem-solving systems).

The HSII Problem-Solving Organization: A Production System Approach

The HSII system organization, which can be characterized as a "parallel" production system, has a centralized data base which represents the dynamic problem solution state. This data base, which is known as the *blackboard*, is a multidimensional data structure which is readable and writable by any precondition or knowledge-source process (where a knowledge-source process is the embodiment of a production action).[1] Preconditions are procedurally oriented and may specify arbitrarily complex tests to be performed on the data structure in order to decide precondition satisfaction.

[1] As an example, the dimensions of the HSII speech-understanding system data base are informational level (e.g., acoustic level, phonetic level, and word level), utterance time (speech time measured from the beginning of the input utterance), and data alternatives (where multiple hypotheses are permitted to exist simultaneously at the same level and utterance time). For additional details, see Appendix A.

Preconditions are themselves data-directed in that they are tested for satisfaction whenever relevant changes occur in the data base;[1] and simultaneous precondition satisfaction is permitted. Testing for precondition satisfaction is not presumed to be an instantaneous or even an indivisible operation, and several such precondition tests may proceed concurrently.

The knowledge-source processes representing the production actions are also procedurally oriented and may specify arbitrarily complex sequences of operations to be performed upon the data structure. The overall effect of any given knowledge-source process is usually either to hypothesize new data which is to be added to the data base or to verify (and pe haps modify) data previously placed in the data base. This follows the general *hypothesize-and-test* problem-solving paradigm wherein hypotheses representing partial problem solutions are generated and then tested for validity; this cycle continues until the verification phase certifies the completion of processing (and either the problem is solved or failure is indicated). The execution of a knowledge-source process is usually temporally disjoint from the satisfaction of its precondition; the execution of any given knowledge-source process is not presumed to be indivisible; and the concurrent execution of multiple knowledge-source processes is permitted. In addition, a precondition process may invoke multiple instantiations of a knowledge source to work on the different parts of the blackboard which independently satisfy the precondition's pattern. Thus, the independent data-directed nature of precondition evaluation and knowledge-source execution can potentially generate a significant amount of parallel activity through the concurrent execution of different preconditions, different knowledge sources, and multiple instantiations of a single knowledge source.

---

[1] In effect, preconditions themselves have preconditions, call them "pre-preconditions." In HSII, knowledge-source preconditions (which correspond to action preconditions in the production system model) may be arbitrarily complex. In order to avoid executing these precondition tests unnecessarily often, they in turn have pre-preconditions which are essentially monitors on relevant primitive data base events (e.g., monitoring for a change to a given field of a given node in the data base, or a given field of any node in the data base). Whenever any of these primitive events occurs, those preconditions monitoring such events are awakened and allowed to test for full precondition satisfaction. These data events are used by the precondition process as pointers to the specific parts of the data base which may now satisfy the pattern the precondition is monitoring for. During the period between when the precondition process has been first awakened and the time it is executed, the monitoring for relevant data base events continues. Thus, a precondition process, when finally executed, may check more than one part of the data base for satisfaction.

The basic structure and components of the HSII organization may be depicted as shown in the message transaction diagram of Figure 1. The diagram indicates the paths of active information flow between the various components of the problem-solving system as solid arrows; paths indicating control activity are shown as broken arrows. The major components of the diagram include a passive global data structure (the *blackboard*) which contains the current state of the problem solution. Access to the blackboard is conceptually centralized in the *blackboard handler* module, [1] whose primary function is to accept and honor requests from the active processing elements to read and write parts of the blackboard. The active processing elements which pose these data access requests consist of *knowledge-source processes* and their associated *preconditions*. Preconditions are activated by a *blackboard monitoring mechanism* which monitors the various write-actions of the blackboard handler; whenever an event occurs which is of interest to a particular precondition process, that precondition is activated. If upon further examination of the blackboard, the precondition finds itself "satisfied," the precondition may then request a process instantiation of its associated knowledge source to be established, passing the details of how the precondition was satisfied as parameters to this instantiation of the knowledge source. Once instantiated, the knowledge-source process can respond to the blackboard data condition which was detected by its precondition, possibly requesting further modifications be made to the blackboard, perhaps thereby triggering further preconditions to respond to the latest modifications. This particular characterization of the HSII organization, while certainly overly simplified, shows the data-driven nature of the knowledge source activations and interactions.

The following sections of this paper will attempt to refine this diagram of the HSII organization by pointing out the difficulties that arise from this oversimplified representation of the organization and by supplementing the various components of this simple diagram to resolve these problems and result in a more complete organization for AI problem-solving in multiprocessing environments. A more complete message transaction diagram for HSII will be presented in a subsequent section.

[1] The blackboard handler module could be implemented either as a procedure which is called as a subroutine from precondition and knowledge source processes, or as a process which contains a queue of requests for blackboard access and modification sent by precondition and knowledge source processes. In the implementation discussed in this paper, the blackboard handler module is implemented as a subroutine.

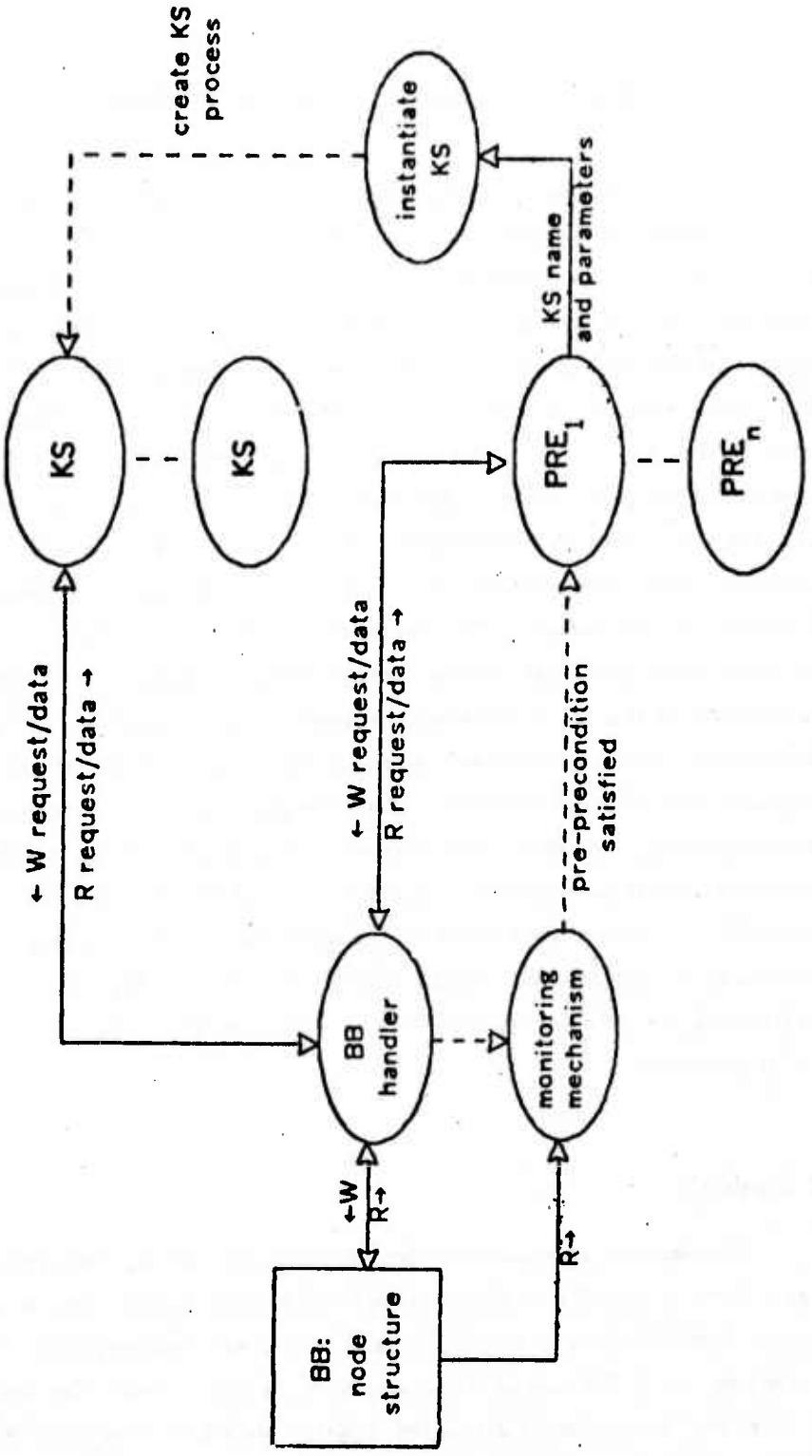


Figure 1. Simplified HSII System Organization

HEARSAY II MULTIPROCESSING MECHANISMS

Given the decision that multiple preconditions may be simultaneously satisfied and that multiple knowledge-source processes may execute concurrently, various mechanisms must be provided to accommodate such a multiprocessing environment. Mechanisms must be provided to support the individual localized executions of the various active and ready processes and to keep the processes from interfering with one another, either directly or indirectly. On the other hand, mechanisms must also be provided so that the various active processes may communicate with one another so as to achieve the desired process cooperation. Since the various constituent knowledge sources are assumed to be independently developed and are not to presume the explicit existence of other knowledge sources, communication among these knowledge sources must necessarily be indirect. The desire for a modular knowledge source structure arises from the fact that usually many different people are involved in the implementation of the set of knowledge sources, and, for purposes of experimentation and knowledge source performance analysis, the system should be able to be easily reconfigured with alternative subsets of knowledge sources. This communication takes two primary forms: data base monitoring for collecting pertinent data event information for future use (*local contexts* and precondition activation), and data base monitoring for the occurrence of data events which violate prior data assumptions (*tags* and *messages*). The following paragraphs will discuss these forms of data base monitoring and their relationship to the data access synchronization mechanisms required in a multiprocess system organization.

Local Contexts

Interprocess communication (and interference) among knowledge sources and their associated preconditions occurs mainly via the global data base, as a result of the design decisions involved in trying to maintain process independence. It is therefore not surprising that the mechanisms necessary to bring about the desired process cooperation and independence are based on global data base considerations. The global data base (the *blackboard*) is intended to contain only dynamically current information. Since preconditions (being data-directed) are to be tested for satisfaction upon the occurrence of relevant data base changes (which are historical *data events*), and since neither precondition testing nor action execution (nor the sequential combination of the

two) is assumed to be an indivisible operation, localized data bases must be provided for each process unit (precondition or action) which needs to remember relevant historical data events. These localized databases, called *local contexts* in HSII, which record the changes to the blackboard since the precondition process was last executed or the knowledge source process was created provide personalized operating environments for the various precondition and knowledge-source processes. A local context preserves only those data events[1] and state changes relevant to its owner. The creation time of the local context (i.e., the time from which it begins collecting data events) is also dependent upon the context owner. Any given local context is built up incrementally: when a modification occurs to the global data base, the resulting data event is distributed to the various local contexts interested in such events. The various primitive data modification routines (or node creation routines) are responsible for the distribution of the data events which result from the modification, just as these modification routines are also responsible for sending warning messages to those processes which want to be notified when specific characteristics of a particular node are altered.[2] Thus, the various local contexts retain a history of relevant data events, while the global data base contains only the most current information.

Data Integrity

Since precondition and knowledge-source processes are not guaranteed to be executed uninterruptedly, these processes often need to assure the integrity of various assumptions they are making about the contents of the data base; for should these assumptions become violated due to the actions of an intervening process, the further computation of the assuming process may have to be altered (or terminated). One way to approach the problem of data integrity is to guarantee the validity of data assumptions by disallowing intervening processes the ability to modify (or perhaps even to examine) critical data. In order to guarantee the integrity of data through the mechanism of exclusive access, the HSII system provides two forms of locking primitives, *node-* and *region-locking*. Node-locking guarantees exclusive access to an explicitly

---

1 The information which defines a data event consists of the locus of the event (i.e., a data node name and a field name within that node) and the old value of the field (the new value being stored in the global data base).

2 The use of these warning messages as way of preserving data integrity will be discussed in the next section.

specified node in the blackboard, whereas region-locking guarantees exclusive access to a collection of nodes that are specified implicitly based on a set of node characteristics. In the current implementation of HSII, the region characteristics are specified by a particular information level and time period of a node. If the blackboard is considered as a two-dimensional structure with coordinates of information-level and time, then region-locking permits the locking of an arbitrary rectangular area in the blackboard. Region-locking has the additional property of preventing the creation of any new node that would be placed in the blackboard area specified by the region by other than the process which had requested the region-lock. Additional locking flexibility is introduced by allowing processes to request read-only access to data fields; this reduces possible contention by permitting multiple readers of a given field to coexist, while excluding any writers of that field until all readers are finished. The system also provides a "super lock," which allows an arbitrary group of nodes and regions to be locked at the same time. A predefined linear ordering strategy for non-preemptive data access allocation (Coffman, *et al.*, 1971) is applied by the "super lock" primitive to the desired node- and region-locks so as to avoid the possibility of data base deadlock.

However, this technique of guaranteeing data integrity through exclusive access is only applicable if all the nodes and regions to be accessed and modified are known ahead of time. The sequential acquisition of exclusive access to nodes and region, without intervening unlocks, can result in the possibility of deadlock. In the HSII blackboard, nodes are interconnected to form a directed graph structure; because it is possible to establish an arbitrarily complex interconnection structure, it is often very difficult for a knowledge-source process to anticipate the sequence of nodes it will desire to access or modify. Thus, the mechanisms of exclusive access cannot always be used to guarantee data integrity in a system with a complex data structure and a set of unknown processes. Further, even if the knowledge source can anticipate the area in the blackboard within which it will work and thereby request exclusive access to this area, the area may be very large, thus leading to a significant decrease in potential parallel activity caused by other processes waiting for this locked area to become available.

An alternative approach to guaranteeing data integrity is to provide a means by which a process (precondition or knowledge source) may place data assumptions about the particular state of a node or group of nodes in the data base (the action of putting these assumptions in the blackboard is called *tagging*). If these assumptions are

invalidated by a subsequent blackboard modification operation of another process, then a *message* indicating this violation is sent to the process making the assumption. In the meantime, the assuming process can proceed without obstructing other processes, until such time as it intends to modify the data base (since data base modification is the only way one process can affect the execution of another). The process must then acquire exclusive access to the parts of the data base involved in its prior assumptions (which parts will have been previously *tagged* in the data base to define a *critical data set*)[1] and check to see whether the assumptions have been violated (in which case, messages indicating those violations would have been sent to the process). If a violation has occurred, the assuming process may wish to take alternative action; otherwise, the intended data base modifications may be made as if the process had had exclusive access throughout its computation. This tagging mechanism can also be used to signal the knowledge-source process that the initial conditions in the blackboard (i.e., the precondition pattern) that caused the precondition to invoke it have been modified; this is accomplished by having the precondition tag these initial conditions on behalf of the knowledge-source process prior to the instantiation of the knowledge source.

In summary, the HSII organization provides mechanisms to accomplish both of these forms of data integrity assurance: the various data base locking mechanisms described previously provide several forms of exclusive or read-only data access; and the data tagging facility allows data assumptions to be placed in the data base without interfering with any process' ability to access or modify that area of the data base (with data invalidation warning messages being sent by data base monitors whenever the assumptions are violated).

To provide a basis for the discussion in the subsequent sections of this paper, Figure 2, depicting the various components of the HSII organizational structure, is offered. The diagram is a more detailed version of the message transaction model presented previously. The new components of this diagram are primarily a result of addressing multiprocessing considerations.

As in the earlier, more simplified organizational diagram, the dynamically current state of the problem solution is contained in a centralized, shared data base, called the *blackboard*. The blackboard not only contains data *nodes*, but it also records

[1] Actually, the requirement is that no other process be able to write to these parts of the data base.

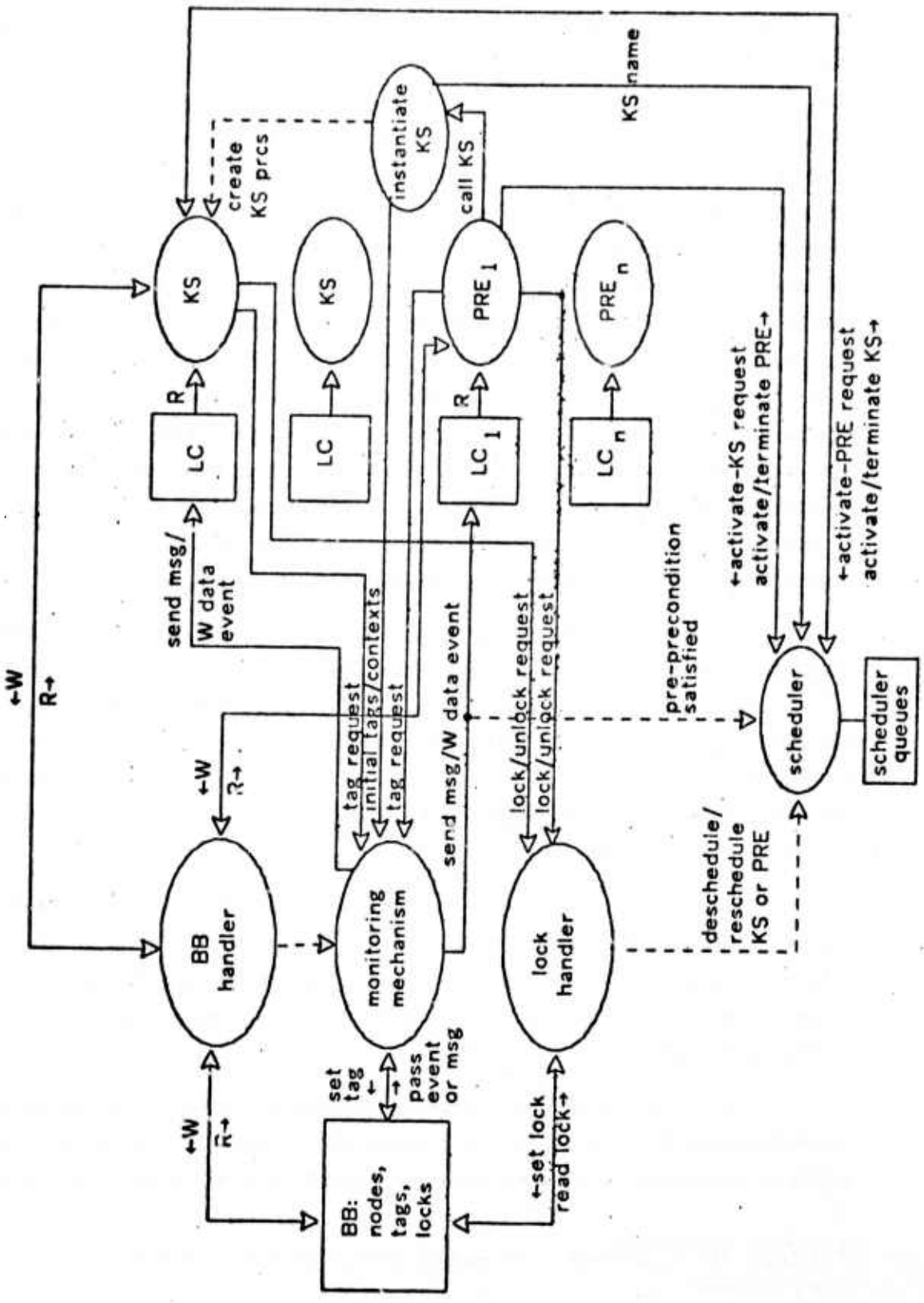


Figure 2. HSII System Organization

data monitoring information (*tags*) and data access synchronization information (*locks*). Access to the blackboard is conceptually centralized in three modules. As before, the *blackboard handler* module accepts and honors read and write data-access requests from the active processing elements (the *knowledge-source processes* and their *precondition processes*). A *lock handler* coordinates data-access synchronization requests from the knowledge-source processes and preconditions, with the ability to block the progress of the requesting process until the synchronization request may be satisfied. A *monitoring mechanism* is responsible for accepting *data tagging* requests from the knowledge-source processes and preconditions, and for sending *messages* to the tagging processes whenever a tagged data field is modified. It is also the responsibility of the monitoring mechanism to distribute *data events* to the various *local contexts* of the knowledge-source processes and preconditions, as well as to activate precondition processes whenever sufficient data events of interest to those preconditions have occurred in the blackboard.

Associated with each active processing element is a local data base, the *local context*, which records data events that have occurred in the blackboard and are of interest to that particular process. The local contexts may be read by their associated processes in order to find out which data nodes have been modified recently and what the previous values of particular data fields were. The local contexts are automatically maintained by the blackboard monitoring mechanism.

Upon being activated and satisfied, precondition processes may in[illegible]e a knowledge source (thereby creating a *knowledge-source pro*[illegible] passing along the reasons for this instantiation as parameters to the new knowle[illegible]e-source process and at the same time establishing the appropriate data monitoring connections necessary for the new process. The *goal-directed scheduler* retains the actual control over allocating hardware processing capability to those knowledge-source processes and precondition processes which can best serve to promote the progress of the problem solution.[1]

[1] One way a scheduler might help in reducing (or eliminating) global data base access interference is to schedule to run concurrently only processes whose global data demands are disjoint. Such a scheduling policy could even be used to supplant an explicit locking scheme, since the global data base locking would be effectively handled by the scheduler (albeit probably on a fairly gross level). Of course, other factors may rule out such an approach to data access synchronization, [illegible]ch as an inability to make maximal use of the available processing resources if only data-disjoint processes are permitted to run concurrently, or the inability to know in

EXPERIMENTS WITH AN IMPLEMENTATION

The preceding sections of this paper have presented various of the mechanisms necessary in implementing a knowledge-based problem-solving system such as HSII in a multiprocessing environment. The present sections will discuss the various experiments that have been performed in an attempt to characterize the multiprocessing performance of the HSII organization in the speech-understanding task.

HSII Multiprocess Performance Analysis through Simulation

In order to gain insight into the various efficiency issues involving multiprocess problem-solving organizations, a simulation model was incorporated within the uniprocessor version of the HSII speech-understanding system. The HSII problem-solving organization was not itself modeled and simulated, but rather the actual HSII implementation (which is a multiprocessing organization even when executing on a uniprocessor) was modified to permit the simulation of a hardware multiprocessor environment.

There were four primary objectives of the simulation experiments: a) to measure the software overheads involved in the design and execution of a complicated, data-directed multiprocess(or) control structure. b) to determine whether there really exists a significant amount of parallel activity in the speech-understanding task, c) to understand how the various forms of interprocess communication and interference, especially that from data access synchronization in the blackboard, affect the amount of effective parallelism realized, and d) to gain insight into the design of an appropriate scheduling algorithm for a multiprocess problem-solving structure. Certainly, any results presented will reflect the detailed efficiencies and inefficiencies of the particular system implementation being measured, but hopefully the organization of HSII is sufficiently general that the various statements will have a wider quantitative applicability for those considering similar multiprocess control structures.

By way of summary, the primary characteristics of the HSII organization

---

advance the precise blackboard demands of each knowledge-source instantiation. Nonetheless, the information relating to the locality of knowledge-source data references is useful in scheduling processes so as to avoid excessive data access interference (thereby improving the effective parallelism of the system).

include: a) multiple, diverse, independent and asynchronously executing knowledge sources, b) cooperating (in terms of control) via a generalized form of the hypothesize-and-test paradigm involving the data-directed invocation of knowledge-source processes, and c) communicating (in terms of data) via a shared blackboard-like data base in which the current data state is held in a homogeneous, multidimensional, directed-graph data structure.

The HSII Speech Understanding System: The Simulation Configuration

The configuration of the HSII speech-understanding system, upon which the following simulation results were based, consists of eight separate generic knowledge sources (each of which may be realized by several active instantiations at any given moment during the problem solution), each of which represents some body of knowledge relevant to the speech-understanding task. Due to the excessive cost of the simulation effort (and due to the limited stages of development of some available knowledge sources), only a subset of the available knowledge sources was actually used in the simulation experiments. Appendix A (which was extracted from (Lesser, *et al.*, 1974)) contains a more detailed description of the blackboard and the various knowledge sources for the more complete HSII speech-understanding system. The knowledge sources used in the simulation were: the *Segment Classifier*, the *Phone Synthesizer* (consisting of two knowledge sources), the *Phoneme Hypothesizer*, the *Phone-Phoneme Synchronizer* (consisting of three knowledge sources), and the *Rating Policy Module*. These knowledge sources are activated by half a dozen precondition processes (which are permanently instantiated in the system), which are continuously monitoring the blackboard data base for events and data patterns relevant to their associated knowledge sources. Both knowledge sources and preconditions may freely access the centralized blackboard data base, which consists of nine lexicon levels.[1] The particular levels used were chosen so as to facilitate the information exchange between the various component knowledge sources.

This set of knowledge sources and preconditions and the associated operating system facilities provided by the HSII organization were first implemented to execute on

[1] While there are eight conceptual information levels within the HSII speech-understanding system (see Appendix A), the blackboard is abstractly segmented according to *lexicons*, rather than information levels, since lexicons allow a finer abstract decomposition of the blackboard.

a uniprocessor DECsystem-10 computer. The particular implementation represented here was programmed in the Algol-like language, SAIL (Swinehart and Sproull, 1971), using SAIL's multiprocessing facilities (Feldman, *et al.*, 1972) and making extensive use of its LEAP associative data storage facility (Feldman and Rovner, 1969). Thus, while the hardware environment of this version of the HSII speech-understanding system is that of a single processor, the software environment is the multiprocessing structure described throughout this paper. The simulation experiments were then run using this HSII configuration, simulating the hardware environment of a closely-coupled multiprocessor where processors can directly communicate with each other through shared memory. The size of the HSII configuration used in the simulations was about 180K, 36-bit words; 70K of this total was the HSII operating system plus the SAIL runtime routines, 73K was precondition and knowledge source code plus variables, and the remainder (which varied from 20K to 45K depending on the number of processors being simulated and the number of processes being instantiated) represented the blackboard data base plus process activation records and other SAIL working space. The simulations were carried out to determine the efficiencies of the various HSII multiprocessing mechanisms discussed previously, as well as to gain some insight into any problems that might arise in the ensuing implementation of a HSII speech-understanding system for the Carnegie-Mellon C.mmp multiprocessor.[1] The following sections will discuss the results of the various experiments which have been performed using the multiprocessor-simulation version of the HSII speech-understanding system.

Simulation Mechanisms and Simulation Experiments

The various multiprocessor simulation results were obtained by modifying the flow of control through the usual HSII multiprocessing organization to allow simulation scheduling points every time a running process could interact in any way with some other concurrently executing process. Such points included blackboard data base accesses and data base access synchronization points (including attempts to acquire data base resources, both at the system and user levels, and any resulting points of

[1] The implementation of the C.mmp version of the HSII speech-understanding system thus far has been, in fact, essentially a direct mapping of the DECsystem-10 implementation, with additional design being done as necessary to solve the particular problems of running in the C.mmp environment (such as having to resolve the small address space problem, wherein any given process may have at any one moment only a 32K-word window into the centrally located main memory).

process suspension due to the unavailability of the requested resource, as well as the subsequent points of process wake-up for retrying the access request). Simulation scheduling points were also inserted whenever a data modification warning message (triggered by modifying a tagged data field) was to be sent, as well as whenever a process attempted to receive such a message. The scheduling mechanism itself was also modified to allow for the simulated scheduling of multiple processing units, while maintaining the state information associated with each processor being simulated (such as the processor clock time of that simulated processor and the state of the particular process being run on that processor). The simulation runs were performed so as to keep the processor clock-times of each processor being simulated in step with one another (the simulation being *event-driven*, rather than *sampled*), thereby allowing for the accurate measurement and comparison of concurrent events across processors. By selecting the number of processors to be simulated and choosing the usual scheduling parameters and precondition and knowledge-source parameters, a chronological trace of the activity of each process and processor could be obtained. By accumulating statistics during the trace period and by performing various post-processing operations upon this activity trace record, the simulation results presented in the following sections were obtained.

Most of the results presented here were achieved by using a single set of knowledge sources (as described above), with a single speech-data input utterance, keeping the data base locking structure and scheduling algorithms essentially fixed, while varying the number of simulated (identical) processors. Several runs were also performed to test the effects of altering the knowledge-source set, altering the locking structure, and altering the mode of data input (the normal input mode being a utterance-time-ordered introduction of input data which simulates real-time speech input).

Measures of Multiprocessing Overhead: Primitive Operation Timings

Time measurements of various primitive operations were made using a 10-microsecond hardware interval timer. Some of the timed primitive operations (such as those involving simple data base access and modification) were not especially subject to the fact that the problem-solving organization involved multiple parallel processes, whereas others (such as those involving process instantiation and process synchronization) were directly related to the multiprocess aspects of the organization

(and might even be taken in part as overhead when compared to alternative single-process system organizations). The times for the various system operations, as shown in Table 1, should be read as relative values, comparing the multiprocess-oriented operations with the data accessing operations to get a relative feel for the overheads involved in supporting and maintaining the multiprocess organization of HSII. Keep in mind that such time measurements are highly dependent on the particular implementation and can change fairly radically when implemented differently. In fact, a primary use of such timings is in determining operating system bottlenecks so that such code sections can be rewritten in a more optimal way. As a result, some primitive operations reflect execution times which are a result of extensive optimization attempts, while other operations (in particular, the "super lock" operations, *lock!* and *unlock!*) have not yet been subjected to this optimization.

Table 1 gives timing statistics relating to the costs involved in maintaining the shared, centralized blackboard data base. Two sets of statistics are given, one set showing the operation times without the influence of data access synchronization (blackboard locking) and one set with the locking structures in effect. These two sets of times give a quantitative feeling for the cost of data access synchronization mechanisms in this particular implementation of HSII. The figures given include the average runtime cost per operation, the number of calls (in this particular timing run) to each operation (thereby showing the relative frequencies of operation usage), and the percentage of the overall runtime consumed by each operation. With respect to the individual entries, *create.node* is a composite operation (involving many field-writes and various local context updates) for creating blackboard nodes. The *read.node.field* and *write.node.field* operations are used in accessing the individual fields of a node. Note that included in any given field-read or -write operation is the cost of perhaps tagging (or untagging) that particular field (or its node). The various functions of the blackboard monitoring mechanism are contained within the field-write operations. Thus, also included in the field-write operation is the cost of distributing the data event resulting from the write operation to all relevant precondition and knowledge-source process local contexts, as well as the cost of sending tag messages to all processes which may have tagged the field being modified; these additional costs are also accounted for independently in the *send.msgs.and.events* and *notify.sset* table entries. Field-write operations are also responsible for evaluating any pre-preconditions associated with the field being modified and activating any precondition whose pre-

	% total runtime		mean time (ms)		number of calls	
	w/o lock	w/ lock	w/o lock	w/ lock	w/o lock	w/ lock
Blackboard Accessing:						
create.node	6.96	4.15	35.81	50.77	287	287
read.node.field	5.06	15.68	0.31	2.03	23577	25279
write.node.field	14.13	7.75	13.96	18.44	1493	1476
Blackboard Associative Retrieval:						
retrieve	2.72	4.98	25.07	109.45	160	160
get.time.adjacent	9.31	15.33	23.44	92.00	586	586
get.struct.adjacent	3.99	6.31	43.35	163.20	136	136
get.nodes.in.rgn	2.05	0.87	2.98	3.00	1015	1015
Process Handling:						
invoke.ks	5.29	2.30	22.64	23.64	345	342
create.ks.prcs	0.75	0.31	3.21	3.22	345	342
ks.cleanup	8.20	5.24	35.06	53.94	345	342
invoke.pre	0.10	1.04	10.44	10.59	14	14
create.pre.prcs	0.42	0.40	8.53	19.57	72	72
Local Context Maintenance:						
transfer.tags	7.12	2.99	9.12	9.17	1152	1149
delete.all.tags	0.52	0.22	2.01	2.03	383	380
notify.sset	6.52	3.01	2.63	2.92	3665	3626
send.msgs.and.events	4.04	2.12	3.68	4.68	1021	1594
receive.msg	0.36	0.15	1.00	1.01	531	530
read.cset.or.sset	0.11	0.05	0.84	0.84	192	192
Data Access Synchronization:						
lock! (overhead)	---	7.78	---	57.47	---	476
unlock! (overhead)	---	3.22	---	23.78	---	476
lock.node	---	2.32	---	2.94	---	2770
exam.node	---	9.34	---	2.40	---	13675
lock.rgn	---	0.11	---	1.77	---	227
write.access.chk	---	0.41	---	0.98	---	1470
read.access.chk	---	14.45	---	1.60	---	31761

Table 1. Primitive Operation Times

precondition is satisfied. Included in the cost of reading a data field (e.g., *read.node.field*) is the cost of verifying the access right of the calling process to the node being read (which could involve a temporary-locking operation,[1] the cost of which is also given independently in the *lock.node* table entry); this access-right checking cost is also separately accounted for by the *read.access.chk* operation. It should be noted that because most of the mechanisms required to implement a data-directed control structure are embedded in the blackboard write operations, the time to execute a write operation is significantly more expensive than a read operation. However, the actual cost in terms of total run time of implementing a data-directed control structure is comparatively small in the HSII speech-understanding system, because the frequency of read operations is much higher than that of write operations. If this relative frequency for read and write operations holds for other task domains (e.g., vision, robotics), then a data-directed control structure (which is a very general and modular type of sequencing paradigm) seems to be a very reasonable framework within which to implement such tasks.

Additional blackboard operation costs are described in the Associative Retrieval section of Table 1. Associative retrieval is based on specifying partial node descriptions (called *matching prototypes*) which serve as a means of retrieving the set of blackboard nodes fitting that partial description. *Retrieve* represents the various retrieval operations possible using these matching prototypes. Retrieval from the blackboard may also be done by requesting the nodes which are time-adjacent (according to the utterance-time dimension of the speech-understanding blackboard) or structurally adjacent (according to the blackboard graph structure) to a given node (or set of nodes); *get.time.adjacent* and *get.struct.adjacent* perform these operations. Furthermore, retrieval may be done by requesting the set of nodes contained within a certain region of the blackboard (by *get.nodes.in.rgn*).

Table 1 also relates the costs of process handling within HSII. Process invocation and process creation are separated (the former being a request from a precondition or knowledge-source process to the scheduler to perform the latter), and the costs are accounted separately, as in *invoke.ks* and *create.ks.prcs*. *Ks.cleanup* is the

[1] If a process has not previously locked the node to which it desires access and the process does not have any other node locked, then the system will temporarily lock the node for the duration of the single read or write operation, without the process having explicitly to request access to the node.

cost of terminating a knowledge-source process; preconditions never get terminated. The cost of initializing and terminating a knowledge-source process (i.e., *invoke.ks* and *ks.cleanup*) is due to the overheads involved in maintaining local contexts, locking structures, and data base monitoring (tagging), all of which are necessitated by the multiprocess nature of the HSII organization. However, in a relative sense, this is not expensive, since the total overhead associated with process handling amounts to only about 9% of the overall execution time.

Additionally, local context maintenance costs are given in Table 1, since they are also a cost of having asynchronous parallel processes. While individual tag creation and deletion is handled by the primitive field-read and -write operations, tags may be transferred from a precondition to the knowledge source it has invoked via *transfer.tags* and destroyed at termination of a process via *delete.all.tags*. As noted above, *notify.sset* and *send.msg.and.events* are sub-operations of the field-write operations and represent the cost of distributing data event notifications to all relevant local contexts. *Receive.msg* is the operation used by precondition or knowledge-source processes to receive a tagging message (or perhaps wait for one, if one does not yet exist); and *read.cset.or.sset* is the operation for retrieving the information from a local context.

Finally, Table 1 gives the costs associated with the data access synchronization mechanism. *Lock!* and *unlock!* represent the overhead costs of locking and unlocking a group of nodes specified by the process requesting access rights. These two operations are among the most complex routines in the HSII operating system, the complexity arising from having to coordinate the allocation of data base resources by two independent access allocation schemes (node-locking and region-locking). This coordination is necessary in order to avoid any possibility of data base deadlock by maintaining a homogeneous linear ordering among all data resources (nodes and regions). The costs of *lock!* and *unlock!* do not include the time spent in performing the actual primitive locking operations. The primitive lock costs are given by *lock.node* (lock a node for exclusive access), *exam.node* (lock a node for read-only access), and *lock.rgn* (lock a region for exclusive access). The access-checking operations (*write.access.chk* and *read.access.chk*) are used by the blackboard accessing routines discussed above.

These timing statistics can be used to determine the amount of system overhead incurred in running precondition and knowledge-source processes under the

HSII operating system. The following summary statistics are offered, given as percentages of the total execution time, the percentages being calculated so as to avoid overlapping between categories (as, for example, factoring blackboard reading costs out of blackboard access synchronization):

Blackboard reading	16%
Blackboard writing	4%
Associative retrieval	7%
Internal computations of processes	27%
Local context maintenance	10%
Blackboard access synchronization	27%
Process handling	9%

Another way of viewing these figures is that approximately half of the execution time involves multiprocessor overheads (i.e., local context maintenance, blackboard access synchronization, and process handling). Based on the assumption that this multiprocess overhead is independent of the parallelism factor achieved,[1] then a parallelism factor of 2 or greater is required in order to recover the multiprocess overhead.

Effective Parallelism and Processor Utilization

The problem-solving organization underlying HSII was designed to take maximum advantage of any separability of the processing or data components available within that organization. Knowledge sources were intended to be largely independent and capable of asynchronous execution in the form of knowledge-source processes. Overall system control was to be distributed and primarily data-directed, being based on events occurring in a globally shared blackboard data base. The intercommunication (and interdependence) of the various knowledge-source processes was to be minimized by making the blackboard data base the primary means of communication, thereby exhibiting an indirection with respect to communication similar to the indirect data-directed form of process control. Such a problem-solving organization was believed to be particularly amenable to implementation in the hardware environment of a network of closely-coupled asynchronous processors which share a common memory. Given

[1] This assumption, based on timing statistics from a series of runs with different numbers of processors, seems valid except for the cost of context swapping and process suspension, which depends upon the amount of data base interference and the number of processors.

sufficiently many completely non-interfering processes (i.e., processes which do not interfere in any way with the execution progress of one another), one would expect the achieved parallelism (speed-up) of that set of processes executing on n identical processors to be a factor of n, as compared to the same set of processes executing on a single processor (assuming the same scheduling and multiprocessing overheads). While the HSII organization attempted to allow the various knowledge sources to be as independent as possible, the various processes were to cooperate with one another (primarily via the blackboard data base) in the effort to effect the problem solution.[1] This necessary cooperation (and the various forms of execution interference resulting from it) was expected to result in the achieved parallelism in a multiprocessor environment being somewhat less than the potential parallelism without interference.

Several experiments were run to measure the parallelism achieved in this particular implementation of the HSII problem-solving organization using varying numbers of identical processors. Each of these experiments was run with the knowledge-source set described previously, using the same input data (introduced into the data base so as to simulate real-time speech input), the same blackboard locking structure, and the same scheduling algorithm, while varying the number of (identical) processors. An example of the graphical output produced by the simulation, for the case of eight processors, is displayed in Figure 3. To comment on these activity plots, the "# runnable processes" plot gives the number of processes either running or ready to run at each simulation scheduling point; the "# running processes" plot gives the number of actively executing processes at each scheduling point; the "# ready processes" plot shows the number of processes awaiting assignment to a processor at each scheduling point; and the "# suspended processes" plot gives the number of processes blocked from executing because of data access interference or because they are waiting on the receipt of a tagging message.

Referring to Figure 3c, notice the spiked nature of the ready-processes plot. This is a result of delaying the execution of a precondition (due to the limited processing power available) beyond the point in time at which its pre-precondition is

[1] Note that the size of the HSII blackboard is expected to grow to only several thousand nodes (hypotheses and links), at, say, 25 field entries apiece, depending, of course, on the task domain. Thus, it is assumed (for the purposes of the current investigations, at least) that the blackboard is entirely resident in primary memory; thus, input/output operations are not an issue here, the system being essentially compute-bound.

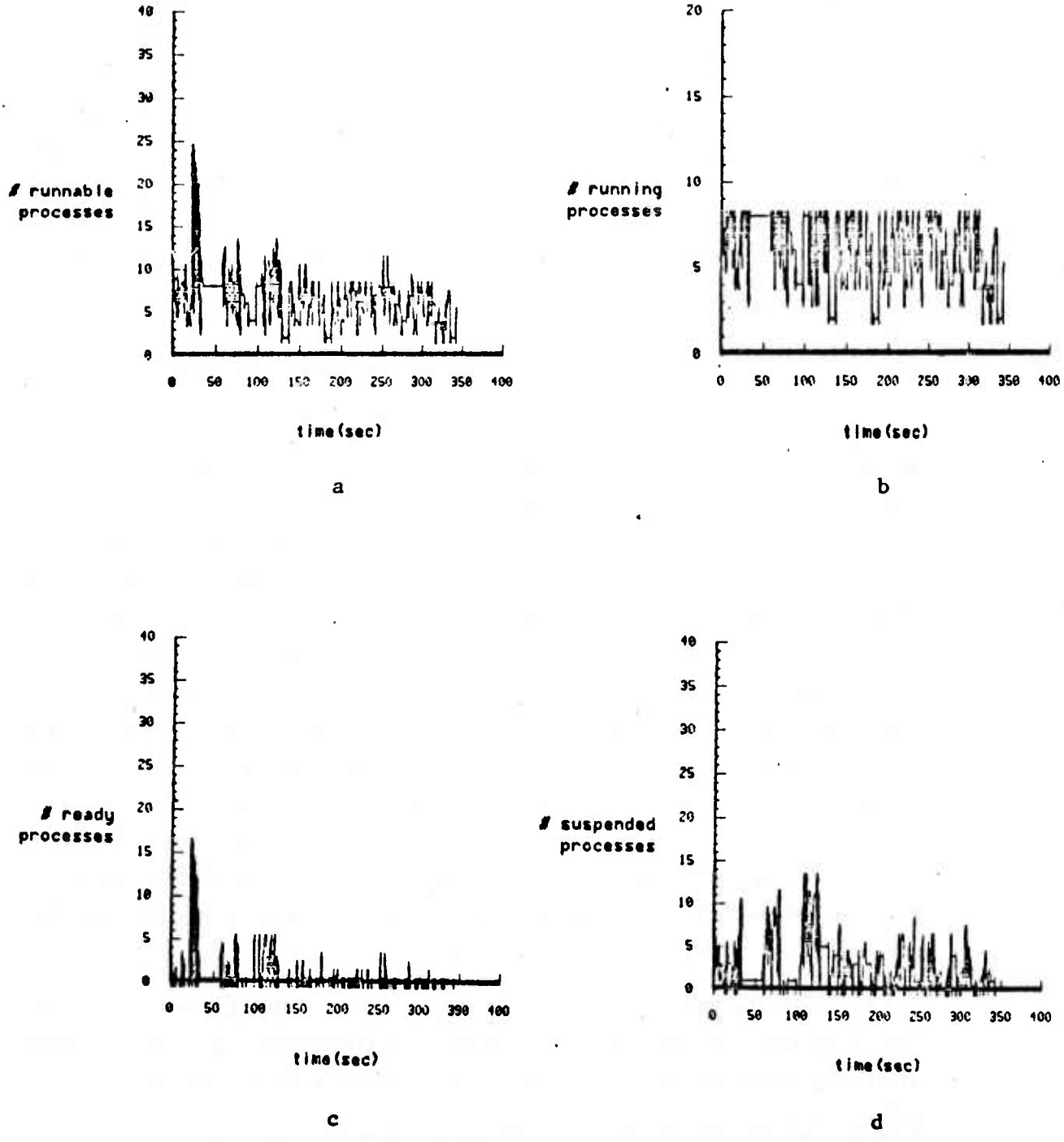


Figure 3a-d. 8 Processors

first satisfied: the longer a precondition is delayed, the more data events it is likely to accumulate in the meantime, and the more knowledge-source processes it is likely to instantiate once it does get executed; hence the spiked nature of the resultant ready-processes plots for configurations of few processors. As parallel processing power increases, preconditions can more often be run as soon as their pre-preconditions are initially satisfied, and the spiking phenomenon subsides.

As an example of how these activity plots have been used in upgrading the performance of the implementation, compare Figure 4 to Figure 3c. Figure 4 depicts the process activity under the control of a scheduler which did not attempt to perform load balancing with respect to ready preconditions; and as a result of not increasing the relative scheduling priority of preconditions as they received more and more data events, the activity spike phenomenon referred to above became predominant, to the extent of reducing process activity to a synchronous system while the long-time waiting precondition instantiates a great many knowledge-source processes all at once.[1] Figure 3c shows the activity on the same number of processors, but using a somewhat more intelligent scheduling algorithm, with a resulting reduction in the observed spiking phenomena. This improved scheduling strategy is the one used for all plots presented herein.

In addition to the plots described above, various other measures were made to allow an explicit determination of processor utilization and effective parallelism for varying numbers of processors. Referring to Table 2, one can get a feeling for the activity generated by employing increasing numbers of processors. All simulations represented in Table 2 were run for equivalent amounts of processing effort with respect to the results created in the blackboard data base by the knowledge source activity. The final clock time of the multiprocessor configuration being simulated is given in simulated real-time seconds, and the accumulated processor idle and lost times are also given. *Idle time* is attributed to a processor when it has no process assigned to it and there are no ready processes to be run; *lost time* is attributed when the process on a processor is suspended for any reason and there are no ready processes

[1] This can be inferred from Figure 4 by noting that the sample points (vertical tick marks) are taken at each simulation scheduling point, and the lack of samples between times 220 and 380 indicates that the process that started running at 220 had no concurrently running processes competing with it until time 380, when there were suddenly 25 new processes contending for computing resources.

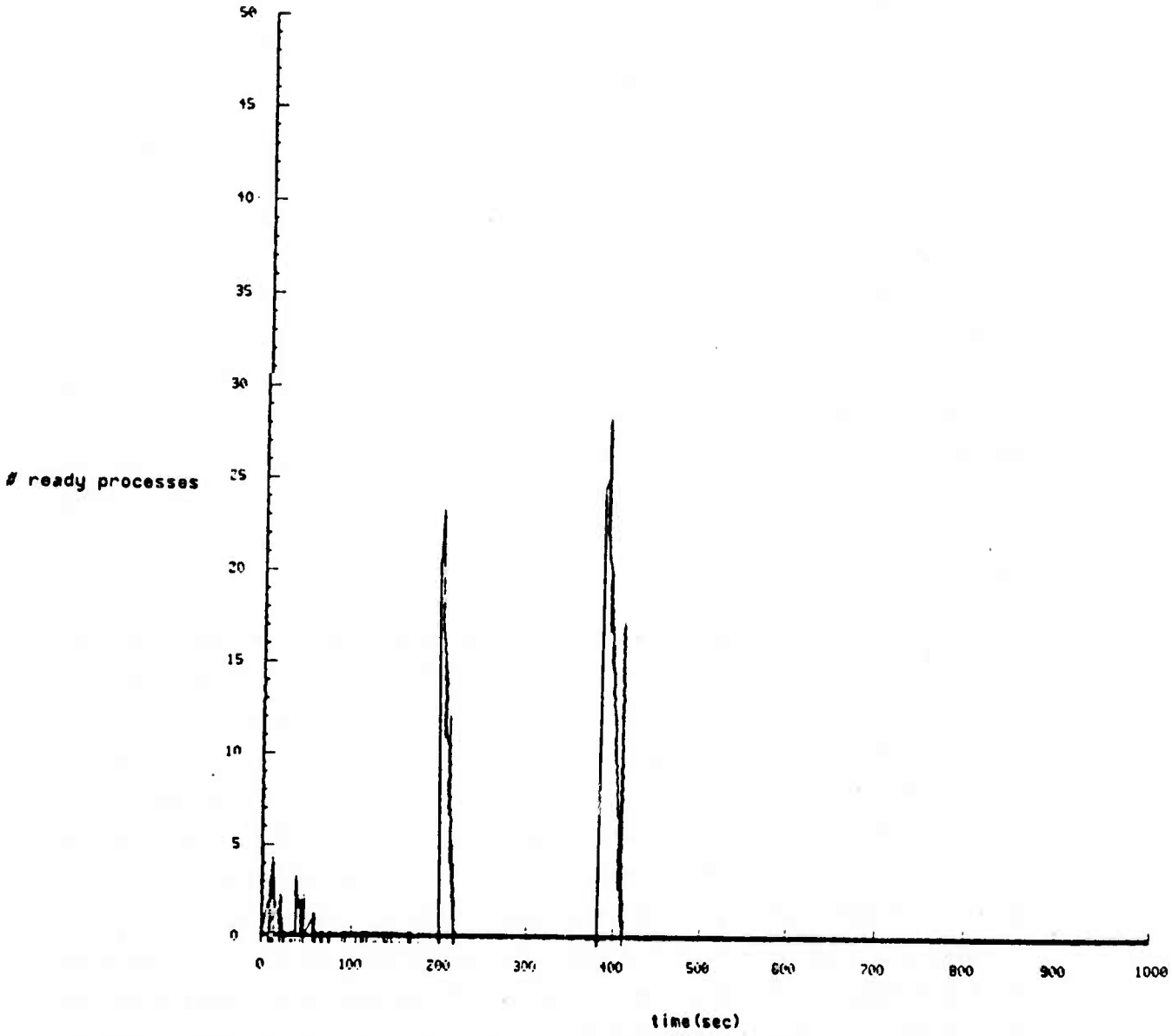


Figure 4. 8 Processors-old scheduling strategy

number of prcrs (all times in secs)	1	2	4	8	16	32 (special*)
KS instantiations	355	401	423	421	415	434
PRE activations	82	126	173	213	200	229
multiprcr clock time	1076	634	389	350	351	43
total idle time	9	15	37	380	2608	867
total lost time	0	5	34	900	1546	0
avg cxt swaps	0	309	942	368	9	0
avg prcr utilization	99%	98%	95%	54%	26%	37%
effective # prcrs	0.99	1.96	3.80	4.32	4.16	11.84
utilization speed-up	1.00	1.98	3.84	4.36	4.20	11.96

* The 32-processor column represents an experiment which was run under special conditions, to be explained below, and should not be compared directly to the other columns of the table.

Table 2. Processor Utilization

which could be swapped in to replace the suspended process. Processor utilization (calculated using the final clock time and processor idle and lost times) is given in Table 2; Figure 5 shows the corresponding effective parallelism (speed-up), based on the processor utilization factors of Table 2.

The speed-up for this particular selection of knowledge sources is appreciable up to four processors, but drops off substantially as one approaches sixteen processors. In fact, a rather distressing feature of this effective parallelism plot is that the speed-up actually decreases slightly in going from eight processors to a sixteen-processor configuration (from a speed-up of 4.36 over the uniprocessor case, down to 4.20). This may be explained by noting that both the eight- and sixteen-processor runs had approximately equal final clock times; but in the sixteen-processor

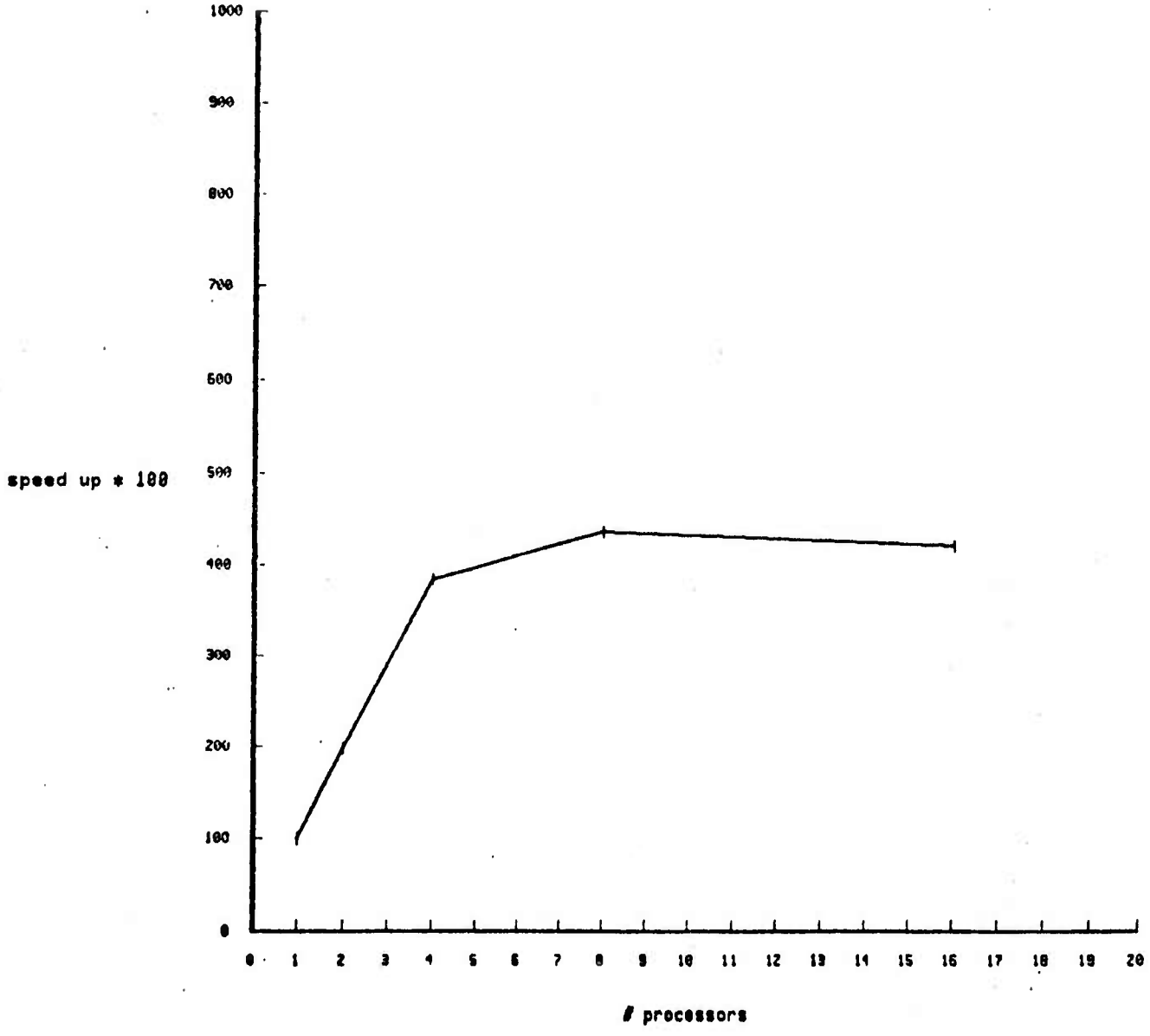


Figure 5. Effective Parallelism According to Processor Utilization

case, the number of runnable processes never exceeded sixteen processes, so any ready process could always be accommodated immediately. As a result, the number of knowledge-source instantiations and precondition activations fell off a bit from the eight-processor case, because the preconditions were more likely to be fully satisfied the first time they were activated (since all ready-processes, knowledge-source processes in particular, could be executed immediately and complete their intended actions sooner, so that when a precondition came to be activated, it would more likely find its full data pattern to be satisfied); thus, preconditions would not often be aborted, having to be re-tested upon receiving a subsequent data event. However, running fewer preconditions resulted in much more idle time for the sixteen-processor configuration (the increase in lost time indicated in Table 2 is an artifact of having too many processors available, since suspended processes would tend to remain on otherwise idle processors rather than being swapped off the processor -- note the rather dramatic decrease in context swaps indicated by Table 2 for the sixteen-processor case). The result is a lower proportionate utilization of the processor configuration, and hence a decrease in the effective parallelism from the eight-processor configuration to the sixteen-processor configuration.

Due to the limited state of development of the total set of knowledge sources, the set of knowledge sources used in the simulation was necessarily limited; so the fact that these plots indicate that not more than about four to eight processors are being effectively utilized is not to say that the full HSII speech-understanding system needs only eight processors. One might ask that if only 4.16 processors of the sixteen-processor configuration are being totally utilized (see Table 2), what is the maximum potential effective parallelism, given this set of knowledge sources? To answer this question, an experiment was performed in which effectively infinite processing power was provided to this knowledge-source set and all data access interference was eliminated (by removing the locking structure overheads and blocking actions); the scheduling algorithm was kept unchanged, as was the input data, although the input data stream was entered so as to be instantaneously available in its entirety (rather than being introduced in a simulated real-time, "left-to-right" manner). The results of this experiment are summarized by the 32-processor column of Table 2 (32 processors was an effective infinite computing resource in this case, since eight of the processors were never used during the simulation). Notice that no lost time was attributed to the run, due to the lack of locking interference; and the resultant processor utilization was 37%

of 32 processors, or 11.84 totally utilized processors. Thus, data base interference caused by particular data base accessing patterns and associated locking structures of the knowledge source set used in the experiment significantly affected processor utilization; if the use of the locking structures could be accomplished in a more non-interfering manner, the speed-up indicated by the eight- or sixteen-processor configurations could be increased substantially. The next section will analyze in detail the exact causes for this data base inteference, and propose changes to the knowledge-source locking structure so as to reduce potential inteference.

Table 3 presents some other system configurations to show effective processor utilizations under varying conditions. The first row repeats the statistics of the sixteen-processor case of Table 2; the second row is a summary of the 32-processor case of Table 2, as described above. Three further data points are offered to indicate the effects of increasing the size of the knowledge-source set. The last three rows of Table 3 involve experiments using an expanded knowledge-source set consisting of the knowledge sources of all the previous runs plus the *Syntactic Word Hypothesizer* (see Appendix A) and its precondition. Using this expanded knowledge-source set, simulations were performed to evaluate the effects of this knowledge-source set on a sixteen-processor configuration with the locking structure in effect, presenting the input data in the usual "left-to-right" manner, as well as in the instantaneous manner used in the infinite-processor test. Comparing the results (in Table 3) to the original sixteen-processor run, the "left-to-right" input scheme achieved a processor utilization of 33%, up 7% from the smaller knowledge-source set case; and by presenting all input data simultaneously, the utilization rose to 35%. The fifth row of Table 3 represents the results of providing effectively infinite computing power (only 25 processors were ever used during the run) to the expanded knowledge-source set and eliminating all data access interference, in the same manner as for the experiment of the second row. In this "optimal" situation for the expanded knowledge-source set, processor utilization was measured at 46%, or 14.72 totally utilized processors. Again, it may be noted that a more effective (less interfering) use of the locking structures can result in substantial increases in processor utilization and effective parallelism.

The addition of the Syntactic Word Hypothesizer was able to achieve the increases in utilization noted in Table 3 because it operates on lexicons that are used by only one other knowledge source (the Phoneme Hypothesizer) in the basic knowledge-source set; hence, the process interference introduced by adding this

experiment description	multiprcr clock	total idle	total lost	% util	effective # prcrs
8 KS's, 6 PRE's 16 prcrs, w/ lock l-to-r input	351	2608	1546	26%	4.16
8 KS's, 6PRE's 32 prcrs, w/o lock instantaneous input	43	867	0	37%	11.84
9 KS's, 7 PRE's 16 prcrs, w/ lock l-to-r input	148	854	726	33%	5.28
9 KS's, 7 PRE's 16 prcrs, w/ lock instantaneous input	155	839	784	35%	5.60
9 KS's, 7 PRE's 32 prcrs, w/o lock instantaneous input	13	226	0	46%	14.72

Table 3. System Configuration Variations

knowledge source was minimal. Unfortunately, the development of knowledge sources at lexicon levels which more directly conflict with those of existing knowledge sources has been limited, so direct experimentation on the interfering effects of such knowledge sources could not be performed; but based on the observations comparing the 32-processor without-lock experiments to the original sixteen-processor with-lock runs, substantial interference due to ineffective use of the locking structure would be expected in such cases of adding "competing" knowledge sources. One mitigating circumstance which could alleviate such interference was noted in the "instantaneous" input case of the expanded knowledge-source set case, as compared to the "left-to-

right" input case: if process activity can be spread across the utterance-time dimension of the blackboard, process interference would decrease -- but interference due to data access synchronization interference can easily overwhelm this improvement. Further experiments along these lines will be attempted as the appropriate knowledge sources become available for use.

Execution Interference Measurements

In addition to the primitive operation timings and achieved parallelism measurements given above, various other measurements were made to determine the various aspects of system performance as related to multiprocessing. As has already been mentioned, a major concern in a multiprocess environment in which the various processes are not entirely independent is that of *execution interference*. Execution interference may arise whenever any given process enters a critical section within which it requires the integrity of a given data structure be maintained (thereby necessitating a means by which to disallow access to others until the critical section is exited). Execution interference may also arise whenever processes must synchronize their activities and perhaps cause themselves to wait on an event based on an action which is to be performed by some external process. Thus execution interference may arise due to causes *external* to the process being delayed (as in the case of trying to access a data structure which is currently held for exclusive access by another process), or the interference may arise due to causes *internal* to the process being delayed (as when a process delays itself by waiting for the occurrence of an externally caused event). As a result of the HSII design philosophy, which states that the various knowledge-source processes should be as independent as possible in specification and execution, most of the execution interference experienced in HSII is of the external variety, wherein a process is delayed due to external causes unknown to itself (and the delay itself is transparent to the process being delayed).

As previously described, there are two methods in the HSII system for preserving data integrity: a) guaranteeing exclusive access through the use of node- and region-locking primitives, and b) placing data assumptions in the blackboard, through tagging primitives, which when violated cause a signal to be sent to the process making the assumption. There is an interesting balance in terms of execution overhead and execution interference between these two techniques. The region-locking

technique is least costly in terms of execution overhead and is the easiest to embed in a program but causes the most execution interference. This is in contrast to the use of tagging which is the most costly in terms of execution overhead and is the most difficult to embed in a program but causes the least execution interference. Both these methods were used for guaranteeing data integrity in the precondition and knowledge-source set that was used in the simulation experiments.

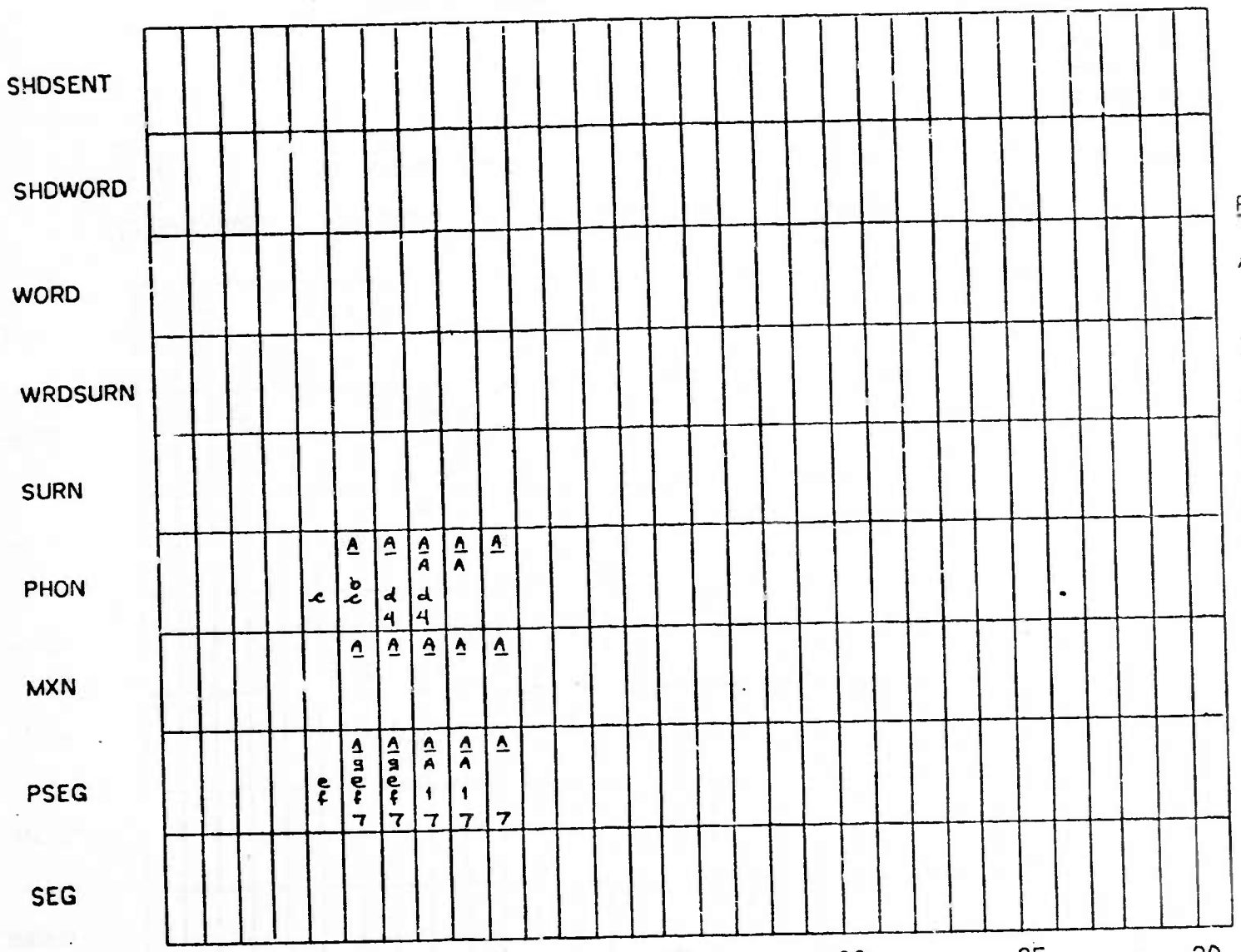
In structuring each knowledge source so as to preserve its data integrity, no *a priori* assumptions were made about the non-modifiability of any blackboard data that knowledge source used in its processing (i.e., it was assumed that any blackboard information that the knowledge source read could perhaps be modified by some other concurrent knowledge-source). This self-contained approach to the design of a knowledge source's locking and tagging structure is required if the modularity of the system, with respect to deletion or addition of knowledge sources, is to be preserved.

The knowledge sources that were used in the simulation experiments were not originally designed so that they could be interrupted at arbitrary points in their processing, and consequently they lacked the appropriate locking and tagging structure to guarantee data integrity in a multiprocess(or) environment. The addition, as an afterthought, of the appropriate locking and tagging structure to these knowledge sources was sometimes quite difficult. This was an especially serious problem when an attempt was made to put tagging primitives into knowledge sources which had internal backtracking control structures for searching the node graph structure in the blackboard. This difficulty arises because previously made data assumptions (tags in the blackboard) associated with a partial path (sequence of nodes in the blackboard) must be removed upon discovering that the path cannot be successfully completed. Thus, most of the knowledge sources in the experiment did not use tagging as a method of guaranteeing integrity, but rather used a combination of node- and region-locking. However, preconditions, which have a much simpler structure and generally do not write in the blackboard, were modified to use the tagging mechanism. In addition, to further simplify knowledge-source locking structures, region-locking was used wherever possible. This excessive use of region-locking was mainly responsible for the significant amount of interference among processes which caused the effective processor utilization to go from an optimal 12 to a realized 4 (see Table 2).

Figure 6 shows an interesting case demonstrating that the indiscriminate use

of region-locking can obstruct the execution progress of many processes and thereby temporarily reduce the effective parallelism of the system. It represents a snapshot of the blackboard locking structure taken during the execution of the simulation. The grid structure represents the two-dimensional abstract data structure, the dimensions being lexicon level and region element number (corresponding to the utterance-time dimension). At the point of each snapshot, the outstanding node and region locks are indicated, as well as the areas requested (but not yet obtained) by suspended processes. The various (non-interfering) tags placed throughout the data base are also indicated. The key indicates the sets of active and suspended processes (the names referring to the precondition and knowledge source names, and the numbers in the names indicating a process instantiation index unique to that particular process). This particular snapshot was taken from the sixteen-processor simulation run with the smaller knowledge-source set. Notice that PSYN263 has locked regions at the PHON, MXN, and PSEG lexicon levels for its exclusive access; the nodes locked by PSYN263 (hypotheses being indicated by H<sequence number>, and links by L<sequence number>) within these regions are those being created by PSYN263, hence the reason for the region locks. Unfortunately, this locking action resulted in the suspension of six other processes awaiting access to parts of the PHON and PSEG lexicon levels which overlap PSYN263's region-locks. Each of these suspended processes is waiting to acquire access-rights to a node in these locked regions; in fact, PRE!PSYN!PSYN and CSEG259 are both waiting on the same node (H141). The diagram also shows the various (non-interfering) tags which were placed on the various nodes at the PHON and PSEG lexicon levels by three of the processes at some previous time. Figure 7, which is another snapshot of locking structure, shows a case where execution interference was not so significant.

The reason the locking structure plots are localized in the lower left-hand corner of the blackboard structure is that the construction of the data base in the speech-processing task is initially left-to-right due to the time-sequential nature of the speech input. Also, the particular set of knowledge sources chosen for use in the simulation experiments happened to be an effectively bottom-up speech recognition system (some of the top-down knowledge sources having not yet been developed to a stable enough state to have been used in the simulations); hence, activity starts in the lower left-hand corner of the blackboard. Further simulations are planned which will work in a combined top-down and bottom-up fashion, thereby increasing the potential



Running Processes

A/1: !PSYN263

Suspended Processes

B/2: PRE!PSC!PSC
C/3: PRE!UTTBOUNDARIES!PSC
D/4: PRE!RPOL!RPOL
E/5: PRE!PSYN!PSYN
F/6: !CSEG259
G/7: !PSYN262

Region Locks

A

Node Locks

A: H150,H151,H152
L101,L102

Region Waits

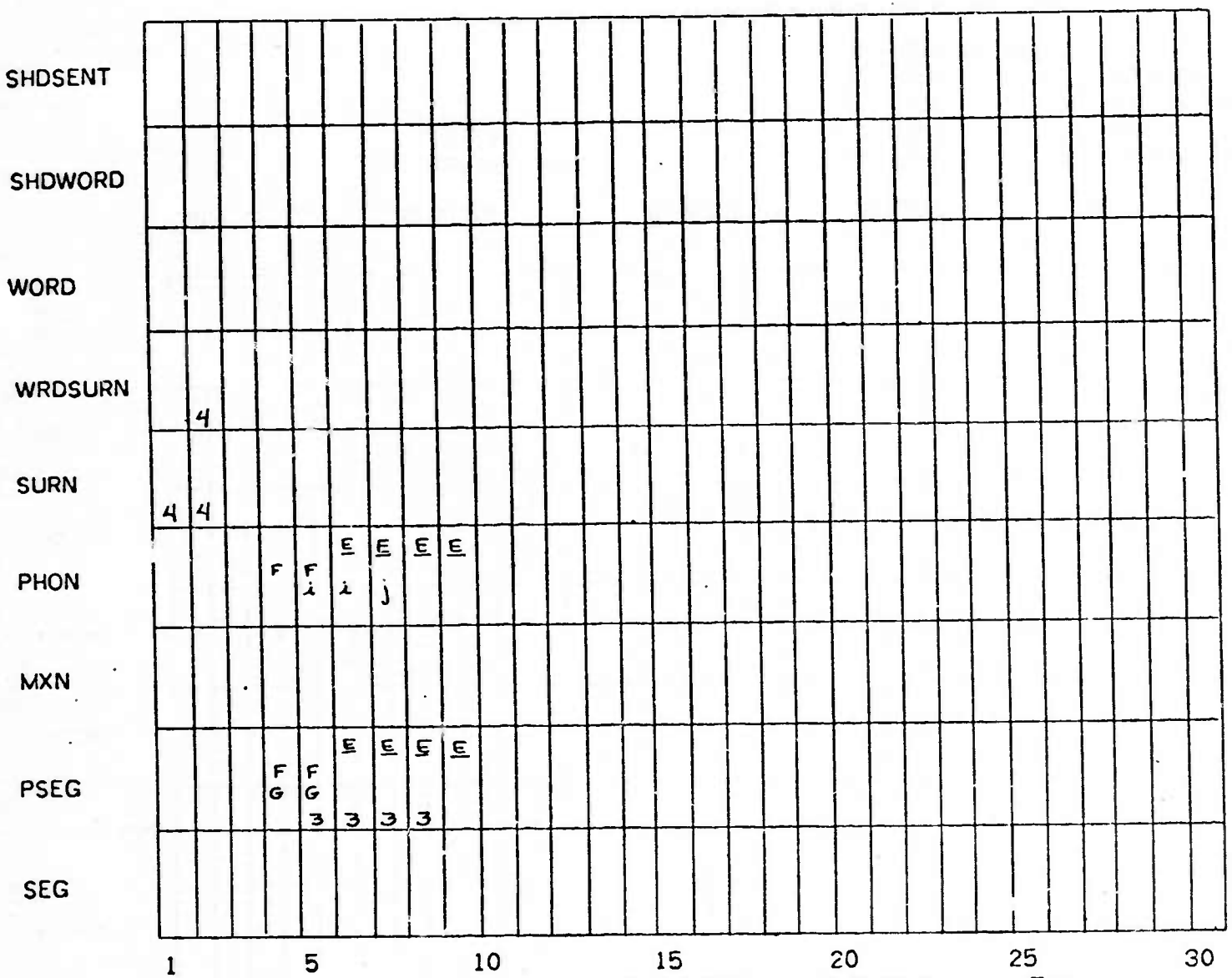
Node Waits

b: H146
c: H142
d: L100
e: H141
f: H141 g: H69

Tags

1: H72,H75,H75,H77
4: H149
7: H69,H70,H70,
H72,H141

Figure 6. 16 Processors- blackboard lock map at time 155.1



Running Processes

A/1: PRE!PSC!PSC
B/2: PRE!RPOL!RPOL
C/3: PRE!PSYN!PSYN
D/4: !SEARCH279
E/5: !TIME280
F/6: !UV282
G/7: !UV283
H/8: !UV284

Suspended Processes

I/9: PRE!UTTBOUNDARIES!PSC
J/10: !TIME281

Region Locks

E

Node Locks

F: H123,H155,L106
G: L107

Region Waits

Node Waits

i: H143
j: H154

Tags

3: H83,H92,H92,H92,
H94,H94,H94,H94,
H98,H98,H98,H98,
H106,H112
4: H27,H27,H38

Figure 7. 16 Processors- blackboard lock map at time 183.1

parallelism (since the top-down knowledge sources will presumably not interfere with the execution of the bottom-up knowledge sources as much as additional competing bottom-up knowledge sources would). The expanded knowledge-source set experiments presented above were a first step in introducing such top-down knowledge; as more knowledge sources become available, their various interference effects will be investigated. Also, other tasks which could use the HSII organization might not necessarily have the left-to-right input characteristics of speech, so future simulations will also test a more distributed input pattern, thereby also increasing the potential parallelism by spreading the processing activity across the breadth of the blackboard; the several experiments presented above which introduced the input in an "instantaneous" manner were the initial attempts in this direction.

A more analytic approach to analyzing the data access interference experienced by precondition and knowledge source processes, for varying numbers of processors, is given in Table 4.

number of prcrs (all times in secs)	1	2	4	8	16
avg BB accesses/KS	54.4	52.8	54.5	53.9	56.4
avg BB accesses/PRE	96.7	68.7	55.7	48.2	51.1
avg prim locks/KS	27.9	27.4	28.0	25.7	26.9
avg prim locks/PRE	96.7	68.7	55.7	48.2	51.1
avg dsched/prim lock(KS)	0	0.020	0.060	0.055	0.053
avg dsched/prim lock(PRE)	0	0.009	0.026	0.045	0.040
avg dsched duration/KS	0	5.08	5.69	1.75	1.90
avg dsched duration/PRE	0	3.95	1.91	1.35	1.86
avg cxt swaps	0	309	942	368	9
avg cxt swaps/dsched	0	1.03	0.97	0.36	0.01

Table 4. Data Access Characteristics

Essentially, Table 4 is an extension of Table 2, which was discussed in the previous section (i.e., the underlying simulation runs were the same for both tables). Execution interference was measured by recording the amount of process suspension (also called *descheduling*), which results from processes being temporarily blocked in their attempts to gain access to some part of the blackboard data base.[1] As might be expected, as process activity increases with increasing numbers of processors, the possibility of execution interference increases (see table entries on "deschedules/primitive lock"). This phenomenon stops at eight processors because in these simulation experiments there were rarely more than eight processes executing at any given moment. At the same time, with more and more processing power available, the likelihood of suspended processes being unblocked and becoming available for further processing increases as the number of processors increases (see table entries on "deschedule duration"). This phenomenon is also indicated by the significant decrease in processor context swaps per deschedule (i.e., with more processors, it becomes less likely that when a process is suspended there will be another process ready to execute).

The major point that can be drawn from this table is that the decrease in processor utilization caused by the locking structure is not due to the high rate of data access interference (i.e., at most only 6% of the primitive locks result in deschedules) but rather from the long duration over which descheduled processes are blocked. This deschedule duration, in the optimal case of 16 processors, where processes do not have to wait for for an available processor, is approximately 2 seconds, which is very close

[1] The number of deschedules attributed to a process is related to the inner workings of the locking mechanism. Not only is the granularity of the locking structure important (i.e., how small a piece of the blackboard data base can be requested for access allocation), but the granularity of the process blocking mechanism is important. For example, processes could be blocked upon trying to gain access to a node and then relegated to waiting in a set of processes which are waiting on <u>any</u> node at the level of the requested node; or the wait set could be divided according to the individual nodes being waited upon. If, in an attempt to conserve semaphore structures, the former strategy is chosen, it could become quite expensive to determine whether, upon receiving an unlock wake-up signal for the wait set, a particular member of the wait set is really re-schedulable as a result of that wake-up signal; hence, it may be cheaper to release all waiting processes in the set, even though all but one will just become descheduled again. If the single-node wait set is used, the costs of maintaining separate semaphores for every possible data object may become prohibitively expensive, although process re-scheduling would not be done unnecessarily in such a scheme.

to the average run time of a knowledge source. This long duration occurs because the knowledge-source locking structures involve executing region locks at the beginning of the knowledge source execution. These region-locks define the entire blackboard area (and perhaps even more) that the knowledge source will either examine or modify during its entire execution.[1] These locks are then released only at the termination of the knowledge source execution. Thus, if data access interference (i.e., a primitive lock deschedule) occurred because of a previously executed region-lock, then the suspended process would very likely not be unblocked until the knowledge source executing the region-lock had completed its processing.

Finally, it is once again admitted that the results presented here are derived from a rather limited selection of knowledge-source processes, the coding style of which may be affected by the various efficiencies and inefficiencies of the particular implementation of the HSII system organization. In particular, since the HSII speech-understanding system is under constant development, various code sections involving the system operations have been subject to extensive optimization attempts, while other sections have not yet had the benefit of such optimization. Additionally, the results are biased by the task domain (viz., speech understanding) and the data structure chosen to represent the dynamic solution state of the task. However, it is hoped that the system organization (including the data base design) is of sufficiently general character that these particular results at least give a feeling for the results that might be expected using a different set of knowledge-source processes to solve the same or different problems.

SUMMARY AND CONCLUSIONS

This paper has presented a design for the organization of knowledge-based AI problem-solving strategies which is felt to be particularly applicable for implementation on closely-coupled multiprocessor computer systems. The method of design is a result of formulating the problem-solving organization in terms of the

[1] Note that the number of primitive lock operations for preconditions is equal to the number of blackboard accesses (from the precondition process averages of Table 4): preconditions do not usually need a long-lasting locked environment (since they do not modify the blackboard except to place tags into it), thus each access is individually protected by the HSII operating system (via temporary-locking), rather than having the precondition perform an explicit *LOCK!* operation before each access.

hypothesize-and-test paradigm for heuristic search, where the various hypothesizers and testers are represented as knowledge sources applicable to the task domain of the problem being solved. A *knowledge source* may be described as an agent that embodies the knowledge of a particular aspect of the problem domain and is useful in solving a problem from that domain by performing actions based on its knowledge so as to further the progress of the overall problem solution. The hypothesize-and-test paradigm provides the conceptual means of coordinating these various knowledge source activities by suggesting that it is the function of some knowledge sources to create *hypotheses* representing a possible (perhaps partial) solution state for the given problem. Hypotheses are created in a global data base and are available for inspection by all knowledge sources. It is the responsibility of other knowledge sources to evaluate these hypotheses in light of their own knowledge of the task domain, and either accept or reject the hypotheses, or propose their own alternative hypotheses (by either modifying the existing hypotheses or creating entirely new ones).

The Hearsay II speech-understanding system (HSII), which has been developed at Carnegie-Mellon University using the techniques for system organization described here, has provided a context for evaluating this system architecture. The HSII organization provides the facilities necessary for knowledge-source cooperation through the hypothesize-and-test paradigm to be carried out in a highly asynchronous and *data-directed* manner, where knowledge sources are specified as independent processing entities capable of parallel execution; the activities of any given collection of such knowledge sources are coordinated by the hypothesize-and-test paradigm through the use of a shared global data base called the *blackboard*.

In specifying the blackboard as the primary means of interprocess communication, particular attention has been paid to resolving the *data access synchronization* problems and *data integrity* issues arising from the asynchronous data access patterns possible from the various independently executing parallel knowledge-source processes. A non-preemptive data access allocation scheme was devised in which the units of allocation could be linearly ordered and hence allocated according to that ordering so as to avoid data deadlocks. The particular units of data allocation (locking) were chosen as being either blackboard nodes (*node-locking*) or abstract regions in the blackboard (*region-locking*). Blackboard nodes also represent the units of data creation within the blackboard. The region-locking mechanism views the potential blackboard as an abstract data space in which access rights to abstract

regions could be granted without regard to the actual data content of these regions.

Another area of concern relating to the use of a shared blackboard-like data facility relates to the assumptions made by the various executing knowledge sources concerning issues of data integrity and localized data contexts. Since the blackboard is intended to represent only the most current global status of the problem solution state, mechanisms were introduced to allow individual knowledge sources to retain recent histories of modifications made to the dynamic blackboard structure in the form of *local contexts*. Knowledge sources are also permitted to mark (*tag*) arbitrary fields (or nodes or regions) of the blackboard itself (without requiring continuing access rights to the field being tagged) and thereby monitor (in a non-interfering way) those locations for subsequent changes; the knowledge source will then be sent *messages* should any modifications be performed upon a tagged field. Local contexts provide knowledge sources with the ability to create a local data state which reflects the net effects of data events which have occurred in the data base since the time of the knowledge source's activation. Combined with the blackboard data tagging capabilities, local contexts also provide a means by which knowledge sources can execute quite independently of any other concurrently executing knowledge sources (and without interfering with the execution progress of any of these processes).

In an attempt to improve the problem-solving efficiency of a multiprocessor implementation of the system by increasing the amount of potential parallelism from knowledge source activity, the logical functions of precondition evaluation and knowledge source execution are split into separate processing entities (called, of course, *precondition* and *knowledge-source processes*). A *precondition process* is responsible for monitoring and accumulating blackboard data events which might be of interest to the knowledge source associated with the precondition; and when the appropriate data conditions for the activation of the knowledge source exist in the blackboard, the precondition will instantiate a *knowledge-source process* based on its associated knowledge source, giving to the new process the data context in which the precondition was satisfied.

The process activity of HSII is intended to be very *data-directed* in nature, basing the decisions as to whether a knowledge source action can be performed on the dynamic data state represented in the blackboard data base. It is the responsibility of a precondition to test this data state for conditions which would warrant the instantiation

of the knowledge source associated with the precondition. The activation of the precondition itself is also data-directed, being based on *monitoring* for the more primitive blackboard modification operations which knowledge-source processes may invoke to effect the results of their computation. This blackboard monitoring is implemented by having the various blackboard modification operators be responsible for the activation of preconditions which are monitoring for data events being caused by the modification operation.

In order to indicate the nature of the performance of the HSII organization when run in a closely-coupled multiprocessor environment, a simulation system was embedded into the multiprocess implementation of HSII on the DECsystem-10. While the results of the simulation are admittedly based on a small (but computationally expensive) set of sample points, they have generally indicated the applicability of this system organization to such a hardware architecture. Given the knowledge-based decomposition of a problem-solving organization as prescribed by the HSII structure, effective parallelism factors of four to six were realized even with a relatively small set of precondition and knowledge-source processes, with indications that up to twelve processors could be totally utilized, given appropriate usage (or structuring) of the data access synchronization mechanisms. Experiments thus far have indicated that careful use of the locking structure is required in order to approach the optimal utilization of any given processor configuration (unless there exist so many ready processes that the number of suspended processes does not matter much, as is the case in configurations of four or fewer processors). An extended use of non-interfering tagging seems to be indicated, along with a reduction in the use of region-locking (perhaps substituting region-examining or node-locking wherever possible). Measurements were also made of various system level primitive operations which are required in order to implement the data-directed multiprocess structure of HSII. While all these results are of a preliminary nature (and hence are subject to variation as various components of the given implementation are improved in their relative efficiencies), they seem to indicate that the HSII organization is indeed applicable for efficient use in a closely-coupled multiprocessor environment.

ACKNOWLEDGMENTS

We wish to acknowledge the contributions of the following people: Lee Erman for his major role in the design and development of HSII, Raj Reddy for many of the basic ideas which have led to the organization described here, and Gregory Gill for his untiring efforts in systems implementation.

SELECTED REFERENCES

Baker, J. (1974). "The DRAGON System -- An Overview," in **Proc. IEEE Symp. Speech Recognition**, Carnegie-Mellon Univ., Pittsburgh, Pa., April 1974, pp. 22-26; also appeared in **IEEE Trans. on Acoustics, Speech, and Signal Processing**, ASSP-23, 1, pp. 24-29 (Feb. 1975).

Bell, C. G., W. Broadley, W. Wulf, A. Newell, et al. (1971). "C.mmp: The CMU Multi-mini-processor Computer," Tech. Rep., Comp. Sci. Dept., Carnegie-Mellon Univ., Pittsburgh, Pa.

Bell, C. G., R. C. Chen, S. H. Fuller, J. Grason, S. Rege, and D. P. Siewiorek (1973). "The Architecture and Application of Computer Modules: A Set of Components for Digital Systems Design," COMPCON 73, San Francisco, Calif.

Coffman, E. G., M. J. Elphick and A. Shoshani (1971). "System Deadlocks," **Computing Surveys** 3, 2, pp. 67-78.

Erman, L. D., and V. R. Lesser (1975). "A Multi-Level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge," 4th International Joint Conference on Artificial Intelligience, Tiblesi, Russia.

Feldman, J. A., and P. D. Rovner (1969). "An Algol-based Associative Language," **Comm. ACM** 12, 8, pp. 439-449.

Feldman, J. A., et al. (1972). "Recent Developments in Sail -- An Algol-based Language for Artificial Intelligence," **Proc. FJCC.**

Fennell, R. D. (1975). "Multiprocess Software Architecture for A.I. Problem Solving," Tech. Rep. (Ph.D. Thesis), Comp. Sci. Dept., Carnegie-Mellon Univ., Pittsburgh, Pa.

Heart, F. E., S. M. Ornstein, W. R. Crowler and W. B. Barker (1973). "A New Minicomputer/Multiprocessor for the ARPA Network," **Proc. AFIPS**, NDD42, pp. 529-537.

Lesser, V. R., R. D. Fennell, L. D. Erman and D. R. Reddy (1974). "Organization of the Hearsay II Speech Understanding System," in **Proc. IEEE Symp. Speech Recognition**, Carnegie-Mellon Univ., Pittsburgh, Pa., April 1974; also appeared in **IEEE Trans. on Acoustics, Speech, and Signal Processing**, ASSP-23, 1, pp. 11-23 (Feb. 1975).

Newell, A. (1973). "Production Systems: Models of Control Structures," in W. C. Chase (ed.) **Visual Information Processing**, Academic Press, pp. 463-526.

Ohlander, R. B. (1975). "Analysis of Natural Scenes," Tech. Rep. (Ph.D. Thesis), Carnegie-Mellon Univ., Pittsburgh, Pa.

Reddy, D. R. (1973a). "Eyes and Ears for Computers," Tech. Rep., Comp. Sci. Dept., Carnegie-Mellon Univ., Pittsburgh, Pa. Keynote speech presented at Conf. on Cognitive Processes and Artificial Intelligence, Hamburg, April, 1973.

Reddy, D. R., L. D. Erman and R. B. Neely (1973b). "A Model and a System for Machine Recognition of Speech," IEEE Trans. Audio and Electroacoust., AU-21, 3, pp. 229-238.

Reddy, D. R., L. D Erman, R. D. Fennell and R. B. Neely (1973c). "The HEARSAY Speech Understanding System: An Example of the Recognition Process," Proc. 3rd Inter. Joint Conf. on Artificial Intel., Stanford, Calif., pp. 185-193.

Swinehart, D. and R. Sproull (1971). SAIL. Stanford AI Proj. Operating Note 57.2, Stanford Univ., Stanford, Calif.

FOCUS OF ATTENTION IN A DISTRIBUTED-LOGIC SPEECH UNDERSTANDING SYSTEM

Frederick Hayes-Roth and Victor R. Lesser
Computer Science Department[1]
Carnegie-Mellon University
Pittsburgh, Pa. 15213

January 12, 1976

ABSTRACT

The Hearsay II speech understanding system under development at Carnegie-Mellon University is a complex, distributed-logic processing system. Processing in the system is effected by independent, data-directed knowledge source processes which examine and alter values in a global data base representing hypothesized phones, phonemes, syllables, words, and phrases, as well as the hypothetical temporal and logical relationships among them. The question of how to schedule the numerous potential activities of the knowledge sources so as to understand the utterance in minimal time is called the "focus of attention problem". Near optimal focusing is especially important in a speech understanding system because of the very large solution space that potentially needs to be searched. Using the concepts of stimulus and response frames of scheduled knowledge source instantiations, competition among alternative responses, goals, and the desirability of a knowledge source instantiation, a general attentional control mechanism is developed. This general focusing mechanism facilitates the experimental evaluation of a variety of specific attentional control policies (such as best-first, bottom-up, and top-down search heuristics) and allows the modular addition of specialized heuristics for the speech understanding task.

1 This research was supported in part by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

INTRODUCTION

The Hearsay II (HSII) speech understanding system (Lesser, et al., 1974; Erman & Lesser, 1975) is a complex, distributed-logic processing system. Inputs to the system are temporal sequences of sets of acoustic segments and associated hypothesized labels. Diverse sorts of speech understanding knowledge are encoded in several (15, currently) independent knowledge source modules (KSs), which include one or more KSs specific to each of the following knowledge domains: acoustic-phonetic mappings, phone expectation-realization relationships, syllable recognition, word hypothesization, and syntax and semantics. The state of processing at any point in time is represented by a global data base (the blackboard) which holds in an integrated manner all of the current hypothesized elements, including alternative guesses, at the various information levels of interpretation (e.g., segmental, phonetic, phonemic, syllabic, word, and phrasal). In addition, any inferred logical or confirmatory relationships among various hypotheses are represented on the blackboard by weighted and directed links between associated hypotheses. The weight and direction of a link reflect the degree to which the hypothesis at the tail of the link implies (supports or confirms) that at the head. The blackboard may be viewed as a two-dimensional problem space, where the time and information level of a blackboard hypothesis serve as its coordinates. Such a view permits consideration of specific "areas" of the problem space and enables us to speak meaningfully of hypotheses in the "vicinity" of a specific data pattern.

Processing in the system consists of additions, alterations, or deletions made to data on the blackboard by the various KSs. Each KS is data-directed, i.e., it monitors the blackboard for arrival of data matching its precondition pattern, a particular pattern of hypotheses and links and specific values of their attributes. Whenever its precondition is matched, the KS is invoked to operate separately on each satisfying data pattern. Finally, when the KS is executed, its (arbitrarily complex) logic is evaluated to determine how to modify the data base in the vicinity of the precondition pattern that triggered the invocation. The data pattern matching the precondition of a KS will be denoted as the stimulus frame (SF) of the invocation, and the changes it makes to the data base as its response frame (RF). Each KS may be schematized as a production rule of the form [precondition => response]. Each instantiation is then schematized [SF => RF], reflecting the fact that the RF data pattern is produced in response to the determination that the SF matches the rule's precondition. Because of the complexity of knowledge source processing, a precise definition of the RF cannot be directly calculated from the stimulus frame without the actual execution of the

knowledge source. However, an abstraction of the RF which specifies the type of changes that may be made (e.g., the addition of a new hypothesis or new link, the modification of a hypothesis' validity, etc.,) and the general vicinity of the changes can be easily calculated directly from the SF. It is this abstraction of the RF which will be used in further discussions.

As is well known in speech understanding research, each KS is imperfect. At any level of analysis, a very large number of errors may be introduced, including misclassifications, failures to recognize, and inappropriate "don't care" responses to what is actually a significant portion of the utterance. The common approach in speech understanding research is to construct systems which can recognize utterances in spite of such errors by evaluating many weakly supported alternative hypothesized interpretations of the speech simultaneously. A practical consequence of this parallel evaluation of numerous alternatives is that, at any point in time, a great number of KS applications are warranted by the existence of hypothetized interpretations matching the various KS preconditions. One object of attentional control is to schedule the numerous potential activities of the KSs to prevent the intractable combinatorial explosion which would inevitably result from an unconstrained application of KSs. More specifically, the focus of attention problem is defined to be that of developing a method for minimizing the total number of KS executions (or total processing time) necessary to achieve an arbitrarily low rate of error in the semantic interpretation of utterances.

The standard approach to the focus of attention problem in other speech systems employing diverse, cooperating sources (Reddy, et al., 1973; Paxton and Robinson, 1975; Woods, 1974) is based on an explicit control strategy. In these explicit control strategies, there is a centralized focusing module which carries out two functions using a built-in set of speech-specific rules: (1) for defining an explicit sequence of calls to a predefined set of knowledge sources and then evaluating their responses in order to determine the suitability of a hypothesized phrase (partial parse of the utterance); and (2) for deciding which of many alternative partial parses of the utterance should be further evaluated. This explicit control strategy is inappropriate in the HSII framework because it destroys the data-directed nature and modularity of knowledge source activity. In the HSII system, KSs can be easily removed or added, and their input and output characteristics changed without effecting other knowledge in the system. There is also a more fundamental argument against an explicit control strategy in a problem-solving system that uses a large number of diverse sources of knowledge: this explicit strategy requires the use of built-in knowledge about the

specific characteristics of knowledge sources. In this case, it seems that the explicit sequential logic necessary to get the appropriate interactions among the knowledge sources in all the possible different data patterns will become very difficult to predetermine and code.

The approach taken in HSII to focus of attention does not use any explicit (pre-compiled) information about which knowledge sources currently are contained in the system, nor their processing characteristics; this approach is more implicit (i.e., mechanistic, uniform, and data-directed); it relies more on general task independent focusing strategies than on speech-specific ones. It should also be noted that, as part of these more general focusing strategies employed in HSII, a uniform mechanism has been incorporated which allows a knowledge source to contribute speech-specific focusing information through modifications to the blackboard. In this way, speech-specific focusing information can be exploited without destroying the modularity and the data-directed nature of knowledge source control in the HSII systems framework.

The remainder of this paper is divided into four sections. In the next section, a number of underlying principles for effective focusing and related processing control mechanisms are described. Subsequently, in the section on "Additional Mechanisms for Precise Focusing," additional objectives for focusing are discussed and related mechanisms for their attainment are presented. The section on "Alternative Policies for Focus of Attention" describes how these techniques permit experimentation with a variety of attentional control policies, such as purely bottom-up, purely top-down, and hybrid analyses. Finally, tentative conclusions are discussed in the last section.

FUNDAMENTAL PRINCIPLES AND MECHANISMS

One can view the focusing problem as a complex resource allocation problem. For example, consider the expenditure of money on alternative search devices in a hunt for oil. The alternative explorers and devices, including seismologists, geologists, drilling teams, and satellite reconnaissance, are the knowledge sources of the task. Each produces its response data only with significant cost and with a substantial probability of error, and there are sequencing constraints which require some KSs to delay their processing until other KSs terminate theirs and then only if particular findings are obtained. How should one invest in their potential contributions? Five fundamental principles have been identified for the control of processing in such tasks, and these are listed below. Each of these principles is used to define a separate

measure for evaluating the importance that should be attached to each KS invocation that has not yet been executed. These measures that are associated with each KS invocation are not necessarily constant for the lifetime of the invocation but may need to be dynamically recalculated as the state of the blackboard changes in the general vicinity of KS's stimulus and response frame. A function based on these measures is then used to associate a priority to each KS invocation.

(1) The competition principle: the best of several alternatives should be performed first. This principle governs how ordering decisions should be made among several behavioral options which are competitive in the sense that a successful outcome of one obviates performing another. For example, consider the problem of determining whether oil exists at site A and suppose that the functions of a geologist and seismologist are substitutable vis-a-vis this objective. If either the seismologist or geologist has already performed and positively indicated the presence or absence of oil, that result obviates employing the other scientist to perform an equivalent function. In this sense, it can be said that the previous result competes with the yet-to-be-performed alternative; that is, the former response is at a higher level of analysis in the same area of the problem space as is the alternative action. However, if oil on site B can be determined only by seismological techniques, hiring a geologist for site A does not compete with hiring a seismologist for site B, according to this principle.

(2) The validity principle: more processing should be given to KSs operating on more valid data. This principle says that, everything else constant, one KS invocation should be preferred to another if the former is working on data which is more credible. In an oil hunt, it would be preferred to employ as a predictor the one seismologist whose seismological readings were most accurate. Similarly, in the speech domain, various KSs will be invoked to contribute to the interpretation of specific data patterns on the blackboard. Each hypothesis in a SF will contain a rating of its validity derived from the validities and implications of hypotheses linked to it. Thus, this principle implies that the KSs invoked to work on the most valid SFs are most preferred. Once these KSs have performed, the hypotheses in their responses will also be rated for validity and will, in general, derive their validity directly from the hypotheses in the SF. By preferring KS invocations with the most credible SFs, the system tends to maximize the validity of its responses.

(3) The significance principle: more processing should be given to KSs whose RFs are more significant. This principle aims at insuring that when a variety of behaviors can be performed, the most important are done first. For example, while

filing a claim on land and drilling are both necessary prerequisites for successful completion of an oil hunt, at the outset of prospecting the former is the more important and should be done first. As an example in the speech domain, a situation might arise where a sequence of phones could be either recognized as a word or subjected to analysis for coarticulation effects. The first of these two actions is more important and, on a priori terms, should be performed first. One heuristic in the speech understanding domain for defining significance is to give preference to KS invocations which are operating at the highest levels of analysis within any portion of the utterance (closest to a complete parse interpretation). A more general statement of this heuristic is that preference should be given to the KS invocation whose RF can potentially produce a result which is closest in terms of information level to the overall goal of the problem solver.

(4) The efficiency principle: more processing should be given to KSs which perform most reliably and inexpensively. Obviously, if one geologist is more reliable than another and the two charge the same for their services, the former should be preferred. Conversely, of two equally reliable geologists, one should prefer the less expensive. Similarly, in the speech domain, many KS applications are more efficient than others and should be preferred. As an example, a bottom-up word hypothesizer is found to be more accurate at generating word hypotheses than is the top-down syntax and semantics KS. Everything else equal, two invocations of these KSs whose response frames consist of new word hypotheses should be scheduled so that the bottom-up hypothesizer is first executed.

(5) The goal satisfaction principle: more processing should be given to KSs whose responses are most likely to satisfy processing goals. The oil hunt managers might establish a goal of determining the depth of water at site A. This would induce additional preference for those agents (e.g., the seismologists and drillers) whose ordinary activities could concomitantly satisfy this additional goal. In the speech domain, similar circumstances arise: the priority of a KS which can potentially generate new word hypotheses in a particular time region of the utterance should be increased. This desire for a specific type of processing is specified in HSII by establishing a goal on the blackboard which represents the time and level of the desired hypotheses. KS instantiations whose RFs match the processing specified in the goal are made more desirable. More generally, KS invocations may be evaluated as more or less likely to help satisfy each specific goal. The higher the probability that a KS invocation will contribute to the satisfaction of a goal and the greater the utility of the goal, the more desirable its execution becomes. Through this mechanism of adding

goals to the blackboard, a knowledge source can dynamically introduce task specific focusing rules into the focusing algorithm. Since KS activity is data-directed, this focusing policy KS would execute only when the data patterns indicating the need for a specific focus action occur.

The preceding five principles provide the theoretical foundation for our attentional control system. A number of sophisticated control mechanisms have been created which provide the tools by which these principles can be converted into operational focusing policies. These mechanisms are discussed in the remainder of this section.

In order to evaluate the preferability of one KS invocation vis-a-vis the others, the five control principles require a number of ordering relationships to hold. In overview, the major operational principle for focusing is to schedule for earliest execution the KS invocation which is the most desirable according to the five rules provided. The focusing mechanism first evaluates the desirability of each KS invocation as a measure of the degree to which it satisfies the various objectives of the system and then executes the most desirable first (with an appropriate generalization for executing several KSs simultaneously in a multiprocessing system). Thus, the major subproblem in the construction of a focuser is the estimation of a KS invocation's desirability. How this desirability is computed will now be described.

Each KS invocation is characterized by a number of attributes. Its SF has a credibility value (between -100 and +100) which estimates the likelihood that the detected pattern of hypotheses and links is valid and satisfies the KS's precondition (negative values imply evidence against this possibility). The credibility value of a SF is determined as a function of the validity ratings on each of the hypotheses in the SF. As previously indicated, these ratings themselves are determined from the strengths of implications on links, the original probabilities assigned to each of the acoustic segment labels provided as input (i.e., the lowest level hypotheses in the blackboard), and the derived validity ratings of intermediate level hypotheses. In our current implementation, the credibility of the SF is taken to be the maximum of the validity ratings of the hypotheses in the SF (ranging from -100 to +100).

Each KS invocation can be thought of as a transformation of the SF into the RF. Associated with the KS invocation then is the estimated level(s) (e.g. phonetic, word, phrasal) of the RF, the estimated validity of the RF hypotheses, and the estimated time (i.e., location and duration) of any newly created RF hypotheses. Each of these estimated values contributes to an appraisal of the significance and probable correctness of the RF which the KS will produce.

The objectives of the significance, efficiency, and goal satisfaction principles can be achieved if the desirability of a KS invocation is computed by any increasing function of the credibility of its SF, the estimated reliability of the KS (to produce correct RFs of the form it anticipates), and the estimated level, duration, and validity of RF hypotheses. The objective of the validity principle, to operate on most valid data first, is accomplished by making desirability an increasing function of the credibility of the SF. The objective of the significance principle, to perform the most significant behaviors first, is achieved by making desirability an increasing function of the level and duration of RF hypotheses. Since hypotheses closest to complete utterance interpretations will be at the highest level and span the entire duration of the speech, actions which can produce such hypotheses or support them will be most preferred. The objective of the efficiency principle, to prefer KSs which perform best, is achieved by making desirability an increasing function of the KSs reliability (per unit "cost" or time).

To understand how the other objectives, the preference of the competition principle for avoiding computation of obviated behaviors and the goal-directed scheduling dictated by the goal satisfaction principle, are achieved in the system, it is necessary to introduce a number of additional concepts. The mechanisms required to operationalize the desired effects of competition will be considered first.

The first objective of the focuser is to insure that the understanding system moves quickly to a complete interpretation of the speech and, in particular, avoids apparently unnecessary computation. Specifically, if any KS invocation is expected to produce a RF which is in the same time range as an existing, higher level, longer duration, and more credible hypothesis, its activity is potentially useless. It is therefore less preferred than the action of a KS which is expected to produce higher level, more expansive, and more credible interpretations of the utterance than those that currently exist. Thus, HSII uses a statistic called the state of the blackboard; this is a single-valued function of each time value, from the beginning of an utterance to its end. The state S(t) for some point (time) t in the utterance is the maximum of the values V(h) of all hypotheses which represent interpretations containing the point t. The value of a hypothesis is an increasing function of its level, duration, and validity. Thus, the highest possible value for a hypothesis would be that associated with the hypothesis representing a complete parse of the entire utterance with a validity rating of +100 (the maximum). To the extent that the utterance is partially parsed in some interval [t1,t2], will the state S(t) be high in this region. Thus, S(t) provides a single metric for evaluating the current success of the understanding process over each area

of the utterance. From a more general viewpoint, the metric V(h) indicates how close a hypothesis h is to the desired overall goal state; and, the metric S measures both what aspect of the overall goal has been solved (e.g., in the case of speech, what time interval) and how good is the solution (e.g., in the case of speech, the validity of the hypothesis and how close in terms of information level it is to the sentential phrase).

It is very easy, using S(t), to decide whether a prospective action is likely to improve on the current state of understanding. If the estimated value V(h) of a RF hypothesis h exceeds S(t) anywhere in the corresponding interval, the KS invocation should be considered very desirable; otherwise it should be inhibited by the existing more valuable, competitive hypotheses. This, in short, is how the objective of the competition principle is accomplished. In addition to its dependence upon the variables already considered, the desirability of a KS invocation is made to be an increasing function of the ratio of the maximum of the estimated value of the RF hypotheses to the current state S(t) (where S(t) is taken to be the minimum over the interval corresponding to the time location of the RF). In this way, preference is given to KS invocations which are expected to improve the current state of understanding.

One can think of S(t) as defining a surface whose height reflects the degree of problem solution in each area. In this conception, operations which would yield results below the surface are undesirable (unnecessary), and those which would raise the surface are preferred.

The last objective to be operationalized is that of the goal satisfaction principle. In general, a goal may specify that particular types of hypotheses are to be created (e.g., create word hypotheses between times t_0 and t_1) or existing hypotheses modified in desired ways (e.g., attempt to reject the hypothesized word "no" between t_3 and t_4 by establishing disconfirming relationships between it and the acoustic data). Two types of adjustments are made to the desirability ratings of KS invocations based on their relationships to such goals. The first case arises when there is direct goal satisfaction, meaning that a KS invocation is a possible candidate for solving a goal because its RF matches the desired attributes of the goal. In this case, the desirability of the KS invocation is increased by an amount proportional to the utility of the goal (the degree to which it is held to be important when it is created).

The second type of effect is the result of indirect goal satisfaction. In this case, a KS invocation does not directly satisfy a goal but apparently increases the probability that it will be solved by producing some result which is held to be partially useful for the achievement of the main goal. Two types of indirect goal satisfying actions can be identified. First, there is goal reduction: a KS invocation generates

subgoals whose solution(s) will entail satisfaction of the original goal. For example, as the result of recognizing the sequence "The (gap) dog," the system might establish a goal for the recognition of an adjective between the two recognized words to replace the gap in understanding. Subsequently, some KS might establish several disjunctive subgoals related to this one, such as goals for recognizing the words "shaggy," "cute," "sleepy," etc. Because the satisfaction of any one of these would constitute satisfaction of the original objective, the KS invocation indirectly satisfies the original goal. Its desirability is less than that of a KS invocation directly satisfying the same goal, but may be more than other KSs.

The second type of indirect goal satisfaction occurs when a KS invocation approaches a goal by producing a RF which is close to the goal but does not quite satisfy it. For example, in the context of the preceding "adjective" goal, a general increase in the activity of knowledge sources which generate and improve phone hypotheses, syllable hypotheses, and phrasal hypotheses in the area of interest will be more or less proximate to the desired response. Since each KS is schematized as a rule of the form [precondition => response], a means-ends analysis can be performed to estimate the probability that some KS invocation will produce a response contributing to the ultimate solution of a goal. The more closely its RF approaches the desired goal, the higher is the probability that execution of a KS invocation will contribute to the goal's ultimate satisfaction and the greater the desirability of the KS invocation.

In summary, the desirability of a KS invocation is defined to be an increasing function of the following variables: the estimated value of its RF (an increasing function of the reliability of the KS and the estimated level, duration, and validity credibility of the hypotheses to be created or supported); the ratio of the estimated RF value to the minimum current state in the time region of the RF; and, the probability that the KS invocation will directly satisfy or indirectly contribute to the satisfaction of a goal as well as the utility of the potentially satisfied goal. Scheduling KS invocations according to their desirabilities then accomplishes the objectives established by the preceding five basic principles. However, there are some inadequacies of such a basic attentional control mechanism; these are considered in the next section.

ADDITIONAL MECHANISMS FOR PRECISE FOCUSING

Basically, while the five fundamental principles appear correct and universally

applicable, they are not complex enough to provide precise control in all of the situations that arise in a complex distributed-logic understanding system. Three additional issues are now introduced, and the control mechanisms currently used to handle these are discussed. The topics considered include dynamically modifiable recognition and output generation thresholds on KS logic; an implicit goal state (approximately the inverse of the current state S(t)) which can be used to determine the desired balance between depth-first and breadth-first approaches to the understanding problem; and methods for avoiding "false peaks" or "cognitive fixedness" in the recognition process.

Nearly all KS behavior can be separated into two components: a pattern recognition component and an output generation component. For example, a word hypothesizer may look for patterns of phones (pattern recognition) in order to produce a new word hypothesis (output generation). Both components operate in fuzzy, errorful ways. In the pattern recognition component, the KS must accept fuzzy matches of its templates because that is the nature of speech recognition. Conversely, the word hypotheses it generates are necessarily probabilistic. The probable correctness of its hypotheses are then reflected by validity ratings or implication weights on its outputs. Thresholding occurs in such processes in two ways. First, the degree of fuzziness tolerated in pattern matching is arbitrarily set to some moderate criterion to prevent an intractably large number of apparent matches. Second, the strengths of the output responses are measured against some threshold to insure that only sufficiently credible responses are produced. The credibility of the response may, in addition to its dependence upon the credibility of the stimulus frame, also be dependent upon the type of inference method used to generate a response. For example, the word recognizer might employ a distance metric for recognition and classification, in which case the credibility of the output word is a decreasing function of the distance between the stimulus phones and the phones of the most similar word template. Responses which are too weak vis-a-vis this second threshold are held in abeyance rather than being produced or forgotten.

Now the general scheme of the robust overall policy that is employed can be sketched. At the beginning of an analysis, relatively high thresholds are specified for pattern matching goodness and output goodness. Processing continues based on the other scheduling principles until thresholds are changed (discussed below). When a threshold change occurs, it may be specific to certain levels or time regions of RFs or to the types of KSs used to produce them. As an example, if all of the utterance were correctly understood except the first word, we would set very low thresholds for

behavior for all KSs in the beginning portion of the utterance. Our current policy, in specific, lowers thresholds most in poorly understood areas adjacent to areas which are well understood. When an arbitrary level of desirability is no longer achieved by any of the pending KS invocations, the important areas for threshold lowering are identified by finding valleys next to peaks in the state function S(t). The thresholds in these areas are lowered in the hope that greater error tolerance there will produce additional results which can be usefully integrated with the adjacent, more reliable interpretations previously produced.

Without dynamically modifiable pattern match and output goodness thresholds, a speech understanding system would necessarily embody numerous parameters whose values were determined at the outset for all problem tasks. Such a system would probably be very sensitive to the particular values chosen. Our approach, however, insures that each of the KSs can be encouraged to perform more work in any area of the blackboard by simply lowering two general sorts of control variables. This is seen as a fundamentally important control principle relating to the controllability of the generative aspect of KSs _per se_ rather than to their comparative expected responses.

The second additional concept which is utilized in the focuser is that of the _implicit goal state_ or I(t). It is only a slight oversimplification to think of I(t) as the inverse of the current state S(t). To the extent that S(t) is large (representing the fact that the portion of the utterance adjacent to t has been highly successfully analyzed), I(t) will be small. A small I(t) value means that there is little to be gained by trying to improve the understanding around t. Conversely, a large I(t) means that the portion of the utterance in the neighborhood of t greatly needs additional analysis. As a result, one might suppose that KSs operating in that region should be conceived as satisfying an implicit goal of raising the level of understanding (the surface of the current state S(t)) wherever it is lowest. In fact, the best role for the implicit goal state is probably as a weak contributor to the desirability of a KS invocation. It remains an empirical question whether it is better to work in the regions of the highest peaks in understanding (_depth-first_) or more evenly throughout the entire utterance (_breadth-first_). Although an optimal strategy is not known, it is clear that in computing the desirability of a KS invocation, the estimated value of the RF and the ratio of the RF value to the minimum of S(t) in the same region are two contributing factors whose relative weightings can be experimentally manipulated to achieve exactly that balance between depth-first and breadth-first which is desired.

As is well known in problem solving and search paradigms, there is a constant danger of getting trapped on "false peaks," as when one bases actions on the apparent

correctness of highly rated but ultimately incorrect interpretations. A number of the preceding focusing principles have been formulated to insure that processing in the region of highly valued hypotheses is facilitated at the expense of other potential actions; a consequence of this paradigm is that the focuser must take precautions to prevent the "cognitive fixedness" which would be apparent if the focuser failed to abandon those paths which lead nowhere. This is done in the focuser in a simple manner. The highest peak in understanding at any point t in the utterance corresponds to the highest valued hypothesis in that region, and its value is just S(t). Thus, stagnation of the understanding process in a region can be detected whenever S(t) fails to increase for a prolonged time. While preference should still be given to the execution of KS invocations working on the surface of S(t) and promising to increase its value, the focuser must conclude that other KS invocations should now become more desirable than they previously seemed, because they at least may improve the analysis in the stagnant area. This is accomplished by increasing the implicit goal state I(t) whenever S(t) is stagnant for a specified length of time. As a result of increasing I(t), KS invocations operating near the surface of S(t) and previously viewed as marginally desirable become sufficiently desirable to be executed. If any one of them succeeds in increasing S(t), I(t) is promptly reset to be the inverse of S(t). However, each time S(t) stagnates for the specified duration, I(t) is again increased. Thus, false peaks are avoided by actually recognizing the behavioral characteristics of cognitive fixedness: as long as the degree of its understanding remains stagnant, it continually increases the desirability of the competing KS alternatives which previously appeared to be suboptimal in the area of stagnation.

ALTERNATIVE POLICIES FOR FOCUS OF ATTENTION

To this point, general principles for focusing and mechanisms to achieve the realization of these principles have been described. However, there still remains a wide variety of policies which can be superimposed upon these mechanisms in a manner consistent with them but prescribing a specific global search strategy to be employed in speech understanding. This flexibility is considered one of the outstanding virtues of the focuser design since it affords the possibility for empirical evaluation of alternative focus of attention policies. In this section, a number of these policies are identified, and it is shown how each of these can be easily effected within our system. Each policy described would be effected by one or more policy modules, a

KS-like program which is activated whenever specific conditions of interest are detected. This will be clarified by the examples below.

Consider the policy which dictates that, whenever possible, understanding is to proceed bottom-up, from the acoustic segments to the phrasal level. Such a policy would be effected as follows. At the outset the policy module would set a goal with infinite positive utility for RFs at the lowest level and a goal with infinite negative utility for RFs at higher levels. When the system became quiescent, the policy module would be reinvoked by the system. Its response would be to modify the goals so that processing at the two lowest levels would be facilitated and all others inhibited. This process would continue until the highest level was facilitated. At any particular point in the analysis, processing would be restricted to several of the lowest levels and would move upward one level at a time as all the potential activity at a lower level had been completed. Similarily, a purely top-down analysis could be controlled in the same way, substituting "highest" for "lowest", etc.

Under ordinary circumstances, using only the mechanisms detailed in the previous sections, a hybrid analysis will occur. While there is increased desirability associated with RFs at the highest levels, it is to be expected that sometimes there will be areas of the utterance where all desirable KS invocations will be at low levels while in other areas they will be primarily at higher levels.

A left-to-right analysis can be accomplished using goals in the same way as for the purely bottom-up or top-down methods. Here, every time quiescence occurs, the processing from the beginning of the utterance to a point further along in time is facilitated. This would continue until the whole utterance was facilitated by a goal. Right-to-left, obviously, is similarly controlled. Note too that "more or less" left-to-right search can be accomplished by specifying less than infinite goal utilities and by defining "quiescence" to mean that the desirabilities of all KS invocations are below some policy threshold for minimally acceptable desirability.

Perhaps one of the most important types of empirical comparisons to be studied is the breadth vs. depth-first alternatives. Breadth-first is, theoretically speaking, advantageous when KSs are capable of looking at broad contexts and optimizing their outputs on the basis of more information than is used, for example, by simple grammatical rewriting rules. Similarly, if KSs are capable of appreciating the extent to which various hypotheses are partially supported by disparate but cooperative data scattered about the blackboard, a breadth-first approach should exhibit some "inteiligence". Alternatively, a depth-first approach is desirable whenever KSs make few errors. For example, if word recognition becomes very good, then it should be

possible to rely upon the words and upon the inferences (e.g., other predicted words) which are derived from them. This reduction in the necessary parallelism of hypothesization makes depth-first a reasonable strategy. In the interim, however, it is apparent that there may be enormous differences in the overall system performance under these different control policies. It is hoped that in the near future empirical data on the relative utility of these different strategies can be obtained. Moreover, if the relative effectiveness of these different control strategies can be associated with formal properties of a problem's structure and complexity, it may be reasonable to anticipate that such empirical observations will be helpful in evaluating the formal complexity of the speech understanding problem.

In summary, it is suggested that the principles and mechanisms described in the preceding sections provide a parameterized framework for the elaboration of numerous alternative "macroscopic" policies for attentional control in the speech understanding problem. Each of the typical sorts of heuristic problem solving policies can be realized by simple policy modules which manipulate goal utilities and respond to quiescence in policy-specific ways.

SUMMARY

By schematizing knowledge sources as [precondition => response] rules, each potential behavior of the Hearsay II system is viewed as an instantiation of such a form. These KS instantiations are seen to be [stimulus frame => response frame] action descriptions. The desirability of an instantiation is then computable from several characteristics of the stimulus and response frames. By enumerating the fundamental principles for attentional control, a desirability measure is produced which handles most of the problems in focusing. Several additional objectives make elaboration of this simple strategy desirable. In order to accomplish more precise overall control, computations are made of the current state of the analysis, the implicit goal state of the system, and the relative degree of goal satisfaction of each KS invocation. Once the desirability of each KS invocation is computed, the execution of the most desirable first serves to accomplish an apparently optimal allocation of computing resources. In addition, our framework provides an excellent environment in which to explore empirically the utility of many global focusing strategies. Each of these can be expressed in terms of particular weightings of the contributions of various terms to the desirability of a KS invocation or by simple modules which create,

modify, and monitor goals which control the direction of analysis. The relatively small grain size of knowledge representation and fine identification of the type and location of knowledge source contributions apparently affords great advantages in constructing mechanisms to control a large, distributed, knowledge-based understanding system.

ACKNOWLEDGMENTS

We would like to acknowledge the help of the following people in the design and implementation of these ideas in the HSII system: Donald Kosy, Craig Everhart, and David McKeown. In addition, Jack Mostow has made numerous contributions to the conceptualization and development of dynamic thresholds which facilitate specific focusing policies.

REFERENCES

Erman, L. D., & Lesser, V. R. A multi-level organization for problem solving using many, diverse, cooperating sources of knowledge. Proc. of the 4th IJCAI, 1975.

Lesser, V. R., Fennel, R. D., Erman, L. D., & Reddy, D. R. Organization of the HEARSAY II speech understanding system. IEEE Trans. on Acoustics, Speech, and Signal Processing, 1975, ASSP-23, 11-23.

Paxton, W. H., & Robinson, A. E. System integration and control in a speech understanding system. A. I. Center, Tech. Note 111, SRI, Menlo Park, Ca. 1975.

Reddy, D. R., Erman, L. D. & Neely, R. D. A model and a system for machine recognition of speech. IEEE Trans. Audio and Electroacoustics AU-21,3, 1973, 229-239.

Woods, W.A. Motivation and overview of BBN SPEECHLIS: an experimental prototype for speech understanding research. Proc. of IEEE Symposium on Speech Recognition, Carnegie-Mellon Univ., Pittsburgh, Pa., 1974, 1-10.

HYPOTHESIS VALIDITY RATINGS IN THE HEARSAY II SPEECH UNDERSTANDING SYSTEM

Frederick Hayes-Roth, Lee D. Erman and, Victor Lesser
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania
January 26, 1976

ABSTRACT

The HEARSAY II speech understanding system under development at Carnegie-Mellon University is a complex, distributed logic processing system. Processing in the system is effected by independent knowledge sources -- data-driven procedures which examine and alter values in a global data base representing hypothesized segments (lowest level), phones, phonemes, syllables, words, and phrases (highest level) as well as the hypothetical temporal and logical relationships among them. Each knowledge source can support a hypothesis unit in two ways: it can assert that the unit is recognizable in terms of lower level hypotheses or that it is predictable with some degree of uncertainty on the basis of previously hypothesized units. These two types of support are called upper and lower implication, respectively. A method for determining the validity rating of a hypothesis based on an initial validity estimate and the current validities and implications of supporting hypotheses is presented.

Hypotheses in Hearsay II represent a variety of types of interpretations of the speech signal, including acoustic segmentation hypotheses, phonetic hypotheses, phoneme hypotheses, syllable hypotheses, word hypotheses, and phrasal (partial parse) hypotheses. Each hypothesis is one of two general types, conjunctive or disjunctive. The conjunctive hypotheses may represent either logical products ("and" relationships) or temporal sequences of lower level hypotheses. The disjunctive hypotheses represent logical summations ("or" relationships) among lower level hypotheses. The degree to which each lower level hypothesis contributes to (supports) the hypothesized relationship is indicated by two numbers, a weight and an implication, which are attached to a link from the lower to the upper hypothesis. The

implications range from -100 (maximally disconfirming) to +100 (maximally confirming). The associated weights range from 0 (least significant) to 100 (most significant). Regardless of its type, each hypotheses may receive predictive support from other hypotheses via links with predictive implications ranging from -100 (maximally counterindicative) to +100 (maximally indicative). In addition, every hypothesis may receive "respelling" support from predicted hypotheses which it may partially or fully realize. For example, a prediction for the class of words referred to as $ME (containing "me" and "us") gives respelling support to any hypothesis of the words "me" or "us" which might realize the expected word class (i.e., which occurs in the same time area as the $ME hypothesis). Respelling implication relationships are reflected by links from the upper to the lower hypothesis and are associated with implications ranging in value from -100 to +100.

A system rating policy module (RPOL) is responsible for determining the validity of each hypotheses as a function of the validities and implications of the hypotheses which support it. Two distinct validity figures are computed. The upper validity (UV) is a measure of the extent to which an hypothesis is supported, ultimately, from the acoustic data. This figure is computed from the hypotheses which support an hypothesis directly from below. The lower validity (LV) is a measure of the extent to which an hypothesis is a plausible prediction of other hypotheses. This figure is computed from the hypotheses which directly predict an hypothesis or respell into it.

The formula for the UV of a disjunctive hypothesis is the maximum of the product of any UV of a supporting hypothesis times the implication from it to the upper hypothesis divided by 100. The formula for the UV of a conjunctive hypothesis is a weighted average of the terms (UV * IMPLICATION / 100) computed for each

hypothesis supporting the hypothesis from below. The significance weights associated with each link determine the relative contribution of each such term in the overall weighted average. In addition, because computing a weighted average tends to cause the UV of hypotheses with many supporting hypotheses to move toward 0, the weighted average of terms is multiplied by a normalizing factor based solely on the number of terms. This factor is 1.0, 1.1, 1.2, 1.3, etc., for hypotheses with 1, 2, 3, ... conjunctive supports.

The LV of an hypothesis is computed as the sum of the terms (UV * PREDICTIVE IMPLICATION) for each hypothesis which predicts it plus the sum of the terms (LV * IMPLICATION / 100) for each hypothesis which gives it respelling implication. This sum is increased or reduced, if necessary, to keep it within the range -100 to +100.

Because the UV is essentially a bottom-up measure of validity and the LV is a top-down measure of validity, some means of combining the two is needed to determine an "overall" hypothesis validity rating. RPOL generates such an overall validity rating by taking 0.9 * UV + 0.5 * LV (restricted to the range -100 to +100). Each knowledge source in the system (including the policy module(s) responsible for scheduling KSs [Hayes-Roth & Lesser, 1976]) can then utilize whichever validity measure is most relevant: UV, LV, or overall validity.

Like other knowledge sources in Hearsay II, RPOL is data-driven. Whenever the validities of hypotheses are modified, new links are created, or implications are changed, RPOL is invoked to recompute whatever validities may have changed. Thus validity changes propagate automatically to all hypotheses immediately or indirectly affected. No cycles occur in such propagation because the UV and LV values are separately computed from bottom-up and top-down supports, respectively. Finally, it

should be noted that hypothesis validity rating is handled in Hearsay II as a system policy function. Knowledge sources contribute to the process only by creating new hypotheses and linking them to pre-existing ones by links whose implications and significance weights completely specify the amount of support which the knowledge source can provide. This uniform method of representation and evaluation means that the effects of changes in the plausibility of any hypothesis are automatically computed by the system, and propagated throughout the data base, without necessitating knowledge source interventions.

ACKNOWLEDGMENT

Don Kosy has provided valuable support to the implementation and maintenance of RPOL.

REFERENCE

Hayes-Roth, F., and Lesser, V. R. Focus of attention in a distributed-logic speech understanding system, 1976. Appears in this volume.

AN AUTOMATICALLY COMPILABLE RECOGNITION NETWORK FOR STRUCTURED PATTERNS

Frederick Hayes-Roth and David J. Mostow
Computer Science Department[1]
Carnegie-Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

A new method for efficient recognition of general relational structures is described and compared with existing methods. Patterns to be recognized are defined by templates consisting of a set of predicate calculus relations. Productions are representable by associating actions with templates. A network for recognizing occurrences of any of the template patterns in data may be automatically compiled. The compiled network is economical in the sense that conjunctive products (subsets) of relations common to several templates are represented in and computed by the network only once. The recognition network operates in a bottom-up fashion, in which all possibilities for pattern matches are evaluated simultaneously. The distribution of the recognition process throughout the network means that it can readily be decomposed into parallel processes for use on a multi-processor machine. The method is expected to be especially useful in errorful domains (e.g., vision, speech) where parallel treatment of alternative hypotheses is desired. The network is illustrated with an example from the current syntax and semantics module in the Hearsay II speech understanding system.

INTRODUCTION

The work described in this paper was motivated by certain problems involved in the task of recognizing general structured patterns and, in particular, the problem of parsing continous spoken speech. From the point of view of the language parser, an essential quality of speech is its errorful nature. Ambiguities in acoustic segmentation, phonetic labelling, word hypothesization, and semantic interpretation necessitate understanding systems which can deal efficiently with multiple alternative hypotheses about each portion of the input. [11] The usual methods of dealing with such multiple hypotheses typically entail an expensive search through a combinatorial space, since they consider only one hypothesis for each portion of input at a time, and then exploit contextual relationships to eliminate certain combinations of adjacent hypotheses as impossible. The data structure and associated recognition procedure described in this paper can be thought of as effectively reversing this process by first exploiting context -- thereby eliminating all but a few combinations from consideration -- and then testing contextually related hypotheses for adjacency. Since the contextual information is statically embedded in the data structure itself, comparatively little work needs to be done at recognition time. This work requires only the computation of a few, simple operations rather than a complex search. Moreover, the method provides an efficient way to handle the spurious insertions and deletions characteristic of speech.

TEMPLATE GRAMMARS

In this section, we define template grammars for recognizing relational structures. A template normal form (TNF) for template grammars is defined. An algorithm described elsewhere [7] translates a given template grammar into an equivalent TNF grammar which is economical in that it maximally exploits repeated subtemplates in the original grammar. The construction of an automatically compilable recognition network (ACORN) from a TNF grammar is described in the next section. The definitions we use are tailored to natural language understanding, but are immediately generalizable to other applications (e.g., vision).

A relation $r(x_1, ..., x_n)$ is an n-ary predicate corresponding to some element or pattern in the language. For example, the relation $\mathbf{tell}(x_1, x_2)$ holds if the word "tell" occurs in the input utterance beginning at time x_1 and ending at time x_2. In general, x_1 and x_2 are temporal arguments specifying the time interval containing a recognized occurrence of the relation, and $x_3, ..., x_n$ are additional attributes of the occurrence. A relation is called primitive if it corresponds to a primitive element (terminal symbol) of the language, non-primitive if it corresponds to a pattern of elements (non-terminal symbol), and top-level if it corresponds to a complete pattern (sentential form).

A template T is a Boolean combination of relations r_i, $i=1, ..., |T|$, restricted as follows. It must be either a disjunction

$$r_1(x_1, ..., x_n) \vee r_2(x_1, ..., x_n) \vee ...$$

$$\vee r_d(x_1, ..., x_n), |T| = d \geq 1,$$

or a conjunction

$$r_1(x_{1_1}, ..., x_{n_1}) \wedge ... \wedge r_p(x_{1_p}, ..., x_{n_p}) \wedge$$

$$\neg r_{p+1}(x_{1_{p+1}}, ..., x_{n_{p+1}}) \wedge ... \wedge$$

$$\neg r_q(x_{1_q}, ..., x_{n_q}), q \geq p \geq 1.$$

In the first case (disjunction), the symbolic arguments $(x_1, ..., x_n)$ are the same for each r_i, $i = 1, ..., d$. In the second case, a weaker condition must be satisfied: the relations must have enough symbolic arguments in common for the template to be connected, that is, for any partition of the p+q relations $r_1, ..., r_{p+q}$ into two non-empty sets A and B, there must exist relations $r_a(x_{1_a}, ..., x_{n_a}) \in A$, $r_b(x_{1_b}, ..., x_{n_b}) \in B$ with $\{x_{1_a}, ..., x_{n_a}\} \cap \{x_{1_b}, ..., x_{n_b}\} \neq \emptyset$.

A template grammar is a set of rules of the form [<template> => <relation>; <action>]. The action optionally associated with each rule specifies what should be done in the event that an instance of the template is recognized in the input and the rule is invoked. Thus a template grammar is actually a production system of the sort described by Newell [12].

Table 1. Sample Grammar (G_{AP})

1. [$\mathbf{Ford}(t_1, t_2)$ => TOPIC(t_1, t_2, expr); expr←"FORD"]

2. [$\mathbf{Rockefeller}(t_1, t_2) \vee \mathbf{Rocky}(t_1, t_2)$ => TOPIC(t_1, t_2, expr); expr←"ROCKEFELLER"]

3. [$\mathbf{Kissinger}(t_1, t_2)$ => TOPIC(t_1, t_2, expr); expr←"KISSINGER"]

1 This research was supported in part by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

4. [or(t_1, t_2) ∧ TOPIC(t_2, t_3, expr) => TOPIC*(t_1, t_3, expr);]

5. [TOPIC(t_1, t_2, $expr_1$) ∧ TOPIC*(t_2, t_3, $expr_2$) =>
TOPIC*(t_1, t_3, expr); expr ← $expr_1$ ∪ $expr_2$]

6. [UTTERANCE(t_1, t_4) ∧ about(t_2, t_3) ∧
TOPIC(t_3, t_4, expr) => TOPIC*(t_3, t_4, expr);]

7. [UTTERANCE(t_1, t_6) ∧ tell(t_1, t_2) ∧ me(t_2, t_3) ∧
nothing(t_3, t_4) ∧ about(t_4, t_5) ∧TOPIC*(t_5, t_6, expr) =>
REJECT(t_1, t_6, expr); SUPPRESS(expr)]

8. [UTTERANCE(t_1, t_6) ∧ tell(t_1, t_2) ∧ me(t_2, t_3) ∧
¬nothing(t_3, t_4) ∧ about(t_4, t_5) ∧TOPIC*(t_5, t_6, expr) =>
REQUEST(t_1, t_6, expr); RETRIEVE(expr)]

As an example, consider the sample template grammar G_{AP} (Table 1) which is part of a much larger grammar for analyzing spoken queries to a wire-service news retrieval system. [4] G_{AP}'s top-level relations are REQUEST and REJECT. An instance of REQUEST is the utterance "Tell me all about Rocky." An instance of REJECT is the utterance "Tell me nothing about Ford, Rockefeller, or Kissinger." The primitive relation UTTERANCE(t_1, t_2), used in rules 6, 7, and 8, simply signifies that the entire utterance spans the time interval [t_1, t_2]; this makes the beginning and ending times of the utterance accessible as arguments to other relations, without violating the framework of the template grammar. Rule 2 illustrates the use of features and actions. The feature, expr, of TOPIC is the semantic expression eventually passed to the actual news retrieval routine. The action of Rule 2 gives expr the value "ROCKEFELLER." Rule 5 is an example of recursion. It handles phrases of the form "$topic_1$, $topic_2$, ..., $topic_{n-1}$, or $topic_n$." The action of Rule 5 forms a compound semantic expression from the expressions associated with its individual constituents. Thus the instance "Ford, Rockefeller, or Kissinger" of the relation TOPIC* has expr = {"FORD", "ROCKEFELLER", "KISSINGER"}. Rule 6 shows how context sensitivity can be embedded in a template grammar. It states that any instance of TOPIC which occurs at the end of an utterance, and whose left context is ABOUT, constitutes an instance of TOPIC*. Rule 8 illustrates the use of negation. It states that any utterance of the form "Tell me ... about X" is a request for information about X unless the gap "..." contains the word "nothing." Thus "Tell me about Ford," "Tell me all about Ford," and "Tell me everything you know about Ford," are all instances of REQUEST. This illustrates the capacity of a template grammar to ignore redundant portions of the input.

A template grammar is in <u>template normal form</u> (TNF) if the following conditions are satisfied:

(1) The template of each rule has one of the following types:

<relation$_1$> ∨ <relation$_2$> ∨ ... ∨ <relation$_d$>, d≥1 (disjunctive type)

<relation$_1$> ∧ <relation$_2$> (conjunctive type)

<relation$_1$> ∧ ¬ <relation$_2$> (negative type)

The relations in a disjunctive template have the same symbolic arguments; the relations in a conjunctive or negative template are connected.

(2) Every non-primitive relation appears on the right side of exactly one rule. Hence we can define the <u>type</u> of a relation to be the type of its unique defining template; a primitive relation is simply said to be of primitive type.

It is clear that any template grammar G can be translated into an equivalent grammar G* in TNF by means of adding new relations and rules. The task of the automatic translator is to do this in such a way as to minimize the number of new relations added. The algorithm we employ is described in [7]. The result of applying the algorithm to the sample grammar of Table 1 is grammar G_{AP}*, shown in Table 2. Mnemonic conventions used in G_{AP}* are these: "+" indicates concatenation; "-" indicates temporal overlap; parenthetical phrases indicate temporal contexts; and "/k" distinguishes different TNF relations arising from occurrences of a single relation in various different rules of the original grammar.

Table 2. Sample Grammar in TNF (G_{AP}*)

1*. [Ford(t_1, t_2) =>
TOPIC/1(t_1, t_2, expr); expr←"FORD"]

2*. [Rockefeller(t_1, t_2) ∨ Rocky(t_1, t_2) =>
TOPIC/2(t_1, t_2, expr); expr←"ROCKEFELLER"]

3*. [Kissinger(t_1, t_2) =>
TOPIC/3(t_1, t_2, expr); expr←"KISSINGER"]

4*. [or(t_1, t_2) ∧ TOPIC(t_2, t_3, expr) =>
TOPIC*/4(t_1, t_3, expr);]

5*. [TOPIC(t_1, t_2, $expr_1$) ∧ TOPIC*(t_2, t_3, $expr_2$)
=> TOPIC*/5(t_1, t_3, expr); expr ← $expr_1$ ∪ $expr_2$]

6**. [UTTERANCE(t_1, t_3) ∧ (ABOUT)TOPIC(t_2, t_3, expr)
=> TOPIC*/6(t_2, t_3, expr);].

7****. [TELL•ME-UTTERANCE-ABOUT•TOPIC*
(t_2, t_3, t_1, t_4, expr) ∧ nothing(t_2, t_3) =>
REJECT(t_1, t_4); SUPPRESS(expr)]

8****. [TELL•ME-UTTERANCE-ABOUT•TOPIC*
(t_2, t_3, t_1, t_4, expr) ∧ ¬ nothing(t_2, t_3) =>
REQUEST(t_1, t_4); RETRIEVE(expr)]

9. [tell(t_1, t_2) ∧ me(t_2, t_3) => TELL•ME(t_1, t_3);]

10. [about(t_1, t_2) ∧ TOPIC*(t_2, t_3) =>
ABOUT•TOPIC*(t_1, t_3);]

11. [TELL•ME(t_1, t_2) ∧ UTTERANCE(t_1, t_3) =>
TELL•ME-UTTERANCE(t_2, t_3, t_1);]

12. [TELL•ME-UTTERANCE(t_2, t_4, t_1)
∧ ABOUT•TOPIC*(t_3, t_4, expr) =>
TELL•ME-UTTERANCE-ABOUT•TOPIC*
(t_2, t_3, t_1, t_4, expr);]

13. [about(t_1, t_2) ∧ TOPIC(t_2, t_3, expr)
=> (ABOUT)TOPIC(t_2, t_3, expr);]

14. [TOPIC/1(t_1, t_2, expr) ∨
TOPIC/2(t_1, t_2, expr) ∨ TOPIC/3(t_1, t_2, expr) =>
TOPIC(t_1, t_2, expr);]

15. [TOPIC*/4(t_1, t_2, expr) ∨
TOPIC*/5(t_1, t_2, expr) ∨ TOPIC*/6(t_1, t_2, expr) =>
TOPIC(t_1, t_2, expr);]

<u>THE RECOGNITION NETWORK</u>

Given a template grammar in TNF, a corresponding recognition network (ACORN), as first described in [6], is constructed as follows. For each relation r appearing in the TNF grammar, there is a unique node, node(r), in the network. (Hence minimizing the number of relations in the TNF grammar is equivalent to minimizing the number of nodes in the network.) For every rule [T => r; A], an arc is drawn from

node(s_i) to node(r) for each relation s_i in the template T. Each node(s_i) is said to be a constituent of node(r), and node(r) a derivative of node(s_i). A node may have zero, one, or more derivatives. The recognition network for the sample grammar G_{AP}, constructed from the [illegible] grammar G_{AP}*, is shown in Figure 1.

Node(r) contains various information: its type (i.e., the type of relation r); the action A in the rule [T => r; A], if any; and the correspondence between the arguments of relation r and the arguments of its constituent relations s_i. This correspondence consists of two parts, a set of tests and a generator. The tests represent any requirements for agreement between the arguments supplied by the constituents node(s_i). The generator is a list of the arguments which are to be supplied in turn to the derivatives of node(r). The arguments are encoded according to a canonical numbering scheme best described by an example. Consider node(TELL•ME). Its constituents are node(TELL), which supplies arguments t_1, t_2, and node(ME), which supplies arguments t_3, t_4. Let L be the concatenated argument list (t_1, t_2, t_3, t_4). Then node(TELL•ME) can specify its arguments by their indices in L. Thus node(TELL•ME)'s only test is L(2) = L(3), denoted by "2:3" below node(TELL•ME) in the network. (See Figure 1.) Similarly, node(TELL•ME)'s generator is the list (L(1), L(4)), denoted by "(1, 4)" above node(TELL•ME) in the network. Arguments which are not supplied by a node's constituents but instead originate at the node itself are specified by negative indices. For example, node(TOPIC/2)'s generator is denoted by "(1, 2, -1)," the -1 specifies the argument expr, which originates at node(TOPIC/2). The action stored in node(TOPIC/2) assigns this argument the value "ROCKEFELLER".

All of the recognition network components described so far are static. There is also associated with each node(r) a dynamic instance list IL. Each instance in the instance list of node(r) represents a single recognized occurrence (instantiation) of the relation r in the input utterance. An instance has several components: a unique identification number I; the time interval [x_1, x_2] containing the occurrence; the values x_3, ..., x_n of additional attributes of the occurrence; and a support set SS containing one or two instance identification numbers. An instance is denoted I:(x_1, ..., x_n; SS). During the recognition process, instances are created and deleted dynamically.

The recognition process is bottom-up, as follows. Initially all instance lists are empty. A lexical analyzer is invoked and begins to scan for occurrences of primitive relations in the input utterance. Since the lexical analyzer receives imperfect, incomplete information from the phonetic labelling routine, the best it can do is to identify possible occurrences. When it finds a possible occurrence of a relation r, it adds a new element to the instance list of node(r) containing the appropriate information. To understand the recognition process, imagine each node(r) as having a demon. The node(r) demon continuously monitors the instance list of each constituent node(s_i) of node(r). Whenever a new instance is added to the instance list of node(s_i), the node(r) demon adds a reference to this new instance to its node(s_i) add set. Similarly, whenever an existing instance of s_i is deleted, the node(r) demon saves a copy of it in its node(s_i) delete set. Add sets and delete sets are referred to collectively as change sets. [9] The demon then activates (wakes) node(r) itself by invoking code pointed to by node(r).

When node(r) is activated, it updates its instance list according to the information in its constituents' instance lists and change sets. If node(r) can derive (construct) any new instances from instances of its constituents, it does so, adding the new instances to its instance list. The support set of each instance contains the identification numbers of the instances from which it has been derived. Node(r) deletes from its instance list any instances supported by (derived from) the defunct instances listed in its constituents' delete sets. The exact way in which all this is done depends, of course, on the type of node(r).

If node(r) is disjunctive, then it has d constituents node(s_1), ..., node(s_d). For each instance I:(x_1, ..., x_n; SS) in a node(s_i)'s add set, node(r) adds a new element I_{new}:(z_1, ..., z_k; {I}) to its own instance list, computing z_1, ..., z_k from the values of x_1, ..., x_n according to the generator stored in node(r). I_{new}'s support set is {I} because the instance I_{new} of r is derived from (supported by, dependent on) the instance I of r's constituent relation s_i. For each defunct instance I in a node(s_i) delete set, node(r) deletes all instances I_{old}:(z_1, ..., z_k; SS) supported by I, i.e., such that I ∈ SS. (Actually, for disjunctive r, all instances of r will have support sets of size one, so I ∈ SS iff SS = {I}. However, for conjunctive r, |SS| = 2; hence the set notation).

If node(r) is conjunctive, then it has exactly two constituents, node(s_1) and node(s_2), with respective instance lists IL_1 and IL_2, add sets AS_1 and AS_2, and delete sets DS_1 and DS_2. First node(r) deletes any of its instances I_{old}:(z_1, ..., z_k; SS) which were derived from instances in DS_1 or DS_2, i.e., those for which SS ∩ (DS_1 ∪ DS_2) ≠ ∅. Then node(r) looks for new instance pairs I_1:(x_1, ..., x_n; SS_1) in IL_1 and I_2:(y_1, ..., y_m; SS_2) in IL_2 such that (x_1, ..., x_n) matches (y_1, ..., y_m) according to the tests stored in node(r). For each such matching pair, node(r) adds a new element I_{new}:(z_1, ..., z_k; {I_1, I_2}) to its instance list, using its generator to select z_1, ..., z_k from x_1, ..., x_n, y_1, ..., y_m. It is sufficient to check only those pairs of instances I_1, I_2 of which one or both are new, or more formally, such that either I_1 ∈ AS_1 and I_2 ∈ IL_2 or I_1 ∈ IL_1 and I_2 ∈ AS_2. For example, suppose the input utterance is "Tell me nothing about Rockefeller," and the lexical analyzer finds an instance I_1:(0, 18; ...) of tell and an instance I_2:(18, 23; ...) of me. Then the test stored in node(TELL•ME) becomes 18 = 18, which is true, so node(TELL•ME) adds a new instance I_{new}:(0, 23; {I_1, I_2}) to its instance list to represent the occurrence of "tell me" in the concatenated time interval [0, 23]. (Time is measured in centiseconds since the beginning of the utterance.) Now suppose the lexical analyzer mistakenly identifies the syllable "fell" in "Rockefeller" as the word "tell," and adds an instance I_3:(257, 269; ...) to node(tell)'s instance list. This may happen, for example, if the phonetic labeller correctly identifies the "F" in "Rockefeller" as an unvoiced consonant but can't tell if it's an "F," a "T," or a "P." No harm is done, however, since when node(TELL•ME) matches I_3 against I_2, the test 269 = 18 fails, and no new instance of TELL•ME is derived from I_3. This example shows how the ACORN automatically weeds out spurious instances hypothesized by the lexical analyzer on the basis of incomplete phonetic information.

Finally, if node(r) is negative, then it has two constituents, node(s_1) and node(s_2), where r = (s_1 ∧ ¬s_2). Let IL_1, IL_2, AS_1, AS_2, DS_1, DS_2 be the instance lists, add sets, and delete sets of node(s_1) and node(s_2). First node(r) deletes any of its instances I_{old}:(z_1, ..., z_k; SS) derived from defunct instances in DS_1, i.e., those for which SS ∩ DS_1 ≠ ∅. Then node(r) looks for any instance pairs I_1:(x_1, ..., x_n; SS_1) in IL_1 and I_2:(y_1, ..., y_m; SS_2) in AS_2 such that (x_1, ..., x_n) matches (y_1, ..., y_m) according to the tests stored in node(r). For each such pair, node(r) deletes all of its instances I_{old}:(z_1, ..., z_k; SS) which depended on I_1, i.e., such that I_1 ∈ SS. This is done since each such I_{old}, previously an instance of (s_1 ∧ ¬s_2), is now invalidated by a new instance of s_2. Adding instances of node(r) is also a bit tricky, and proceeds as follows. First node(r) constructs the set IS of all instances I_1 ∈ IL_1 which match some I_2 in DS_2. Then node(r) looks for all instances I_1:(x_1, ..., x_n; SS_1) in AS_1 ∪ IS which match none of the instances in IL_2. For each such I_1, node(r) adds a new instance I_{new}:(z_1, ..., z_k; {I_1}) to its instance list.

To illustrate this, let us continue with our sample utterance, "Tell me nothing about Rockefeller." Suppose that at some point the lexical analyzer has recognized all the words in the utterance except the word "nothing," and node(TELL•ME-UTTERANCE-ABOUT•TOPIC*) has I_4:(23, 41, 0, 274, "ROCKEFELLER"; ...) on its instance list. Since the instance list of node(nothing) is empty, node(REQUEST) will have an instance I_5:(0, 274, "ROCKEFELLER"; {I_4}) on its instance list. Now suppose that the lexical analyzer finally recognizes the word "nothing," and puts the instance I_6:(23, 41; ...) on node(nothing)'s instance list. This activates both of node(nothing)'s derivatives. Node(REJECT) matches I_6 against I_4, tests 23 = 23 and 41 = 41, and accordingly adds a new instance I_7:(0, 274, "ROCKEFELLER"; {I_4, I_6}) to its instance list. Node(REQUEST) matches I_6 against I_4, tests 23 = 23 and 41 = 41, and accordingly deletes I_5:(0, 274, "ROCKEFELLER"; {I_4}) from its instance list. This example shows how information is accumulated and corrected dynamically during the ACORN recognition process. It also illustrates the ACORN's state-saving nature and its sharing of information between top-level nodes.

Once node(r) has examined its constituents' change sets and, if appropriate, revised its own instance list, it goes back to sleep. Meanwhile, the demons sitting on the derivatives of node(r) have been watching its instance list and, when changes occur, activate their nodes. This chain reaction continues, fuelled by new instances generated by the lexical analyzer, until the lexical analyzer has stopped, all nodes are asleep, and all change sets are empty.

At this point each instance I:(x_1, ..., x_n; SS) of a non-primitive node, node(r), may be interpreted as a partial parse of the interval [x_1, x_2], with relevant syntactic and semantic features given by x_3, ..., x_n. For example, when the recognition of our sample utterance terminates, the instance I:(41, 274, "ROCKEFELLER"; ...) of ABOUT•TOPIC* may be considered to be a partial parse of the input interval [41, 274] containing "about Rockefeller." Parse trees can easily be reconstructed from the information contained in the support sets. Parses of the entire utterance are given by instances of top-level nodes. Thus the instance I_7:(0, 274, "ROCKEFELLER"; {I_4, I_6}) of REJECT constitutes a total parse of the sample utterance, and supplies the semantic feature, expr, required by the action SUPPRESS(expr).

RELATIONSHIP TO EXISTING PARSERS AND PATTERN-MATCHERS

The original motivation which led to the ACORN concept was the development of a general automatic recognition system for spoken utterances, visual scenes, and other structured patterns in which context is a fruitful source of information. Since the speech understanding ACORN treats an utterance as a relational structure, it is related ooth to natural language parsers and to general pattern-matching mechanisms.

The ACORN's closest relative among natural language parsers is PARRY [2], a program which simulates a paranoid individual being interviewed by a psychiatrist. PARRY employs a large library of stored concept sequence templates which are compared with segments of typewritten input sentences. Generalization is achieved by rules which rewrite words as synonymous concepts, delete unrecognized words and, if necessary, delete one recognized word at a time until a template is matched. While the approach underlying PARRY is very successful with typed input, it appears to be too risky for spoken input. Unlike the "perfect" input which PARRY receives, the input to the syntax module of a speech understanding system such as Hearsay II [9] is highly imperfect. PARRY can say, with confidence, "this portion of the input is such-and-such (e.g., the word "oh"), so I'll ignore it;" Hearsay II can only say "if this portion of the input is "oh," I can ignore it; but if it's really the word "no," then I'll need it." An ACORN can be thought of as a non-deterministic version of a PARRY-like system in which all possible parses are followed simultaneously in parallel. On the other hand, an ACORN is capable of recognizing general graph structures and is more powerful than any context-sensitive language parser (string recognizer).

Woods' augmented transition network (ATN) [14] is a mechanism for parsing natural language. It works top-down, uses backtracking, and produces a formal parse of the input sentence. In contrast, an ACORN works bottom-up, does no backtracking, and extracts only those features of the utterance which are relevant to the particular application. An ACORN can be thought of as a state-saving, bottom-up version of an ATN.

Miller [10] has proposed a parser for spoken English which builds multiple partial parse trees and employs a complicated and heuristic search to combine them. An ACORN differs from Miller's parser in handling all combinations simultaneously rather than sequentially, and in the simplicity of the matching operations it uses.

Current artificial intelligence programming systems such as PLANNER [8], QA4 [13], and SAIL [3] can match a given relational template against a data base. However, the method they use is an exhaustive, iterative, and associative search. If several templates are to be matched against the data base, they must be matched one at a time. In contrast, the associative matching operation performed by ACORNs effectively tests all the relations of all the templates simultaneously.

The ACORN's nearest relative among general pattern-matching methods is hierarchical synthesis [1]. Consider the task of matching a template, such as a schematic representation of a building, against an input set of line segments. A recognition algorithm employing hierarchical synthesis replaces the single, many-component template for "building" with a hierarchy of templates for "doors," "windows," "stories," etc. A higher-level template can be matched only if its lower-level constituents are. Hierarchical synthesis considerably reduces recognition time for two reasons. First, it can exploit the repetition of subtemplates by recognizing all instances of a single subpattern just once. Second, before considering whether or not the entire pattern specified by a template is present, it can insure that all necessary subpatterns are present.

However, hierarchical synthesis as described in [1] depends on a hierarchy defined a priori by the user. This limitation is transcended by Hayes-Roth's interference matching method [5], which does hierarchical synthesis in parallel in all possible directions, thereby obviating the need for a predefined hierarchy. In interference matching, a template is represented as a set of relations. Each relation is a predicate with one or more symbolic variables. The input is also a set of relations, whose arguments are constants. A partial match consists of an assignment of input constants to the symbolic variables of a subset of template relations which makes them all true. Interference matching works by finding partial matches and combining them into complete matches.

Like interference matching, the ACORN method is an improved version of hierarchical synthesis in that it requires no predefined hierarchy. The ACORN compiler itself determines an economical hierarchy, and embeds it in the form of a recognition network. Hierarchy selection can be factored out into a separate compilation phase because the choice of hierarchy depends only on the templates and not on the individual input utterance. In interference matching, on the other hand, hierarchy selection depends on the input pattern, and is therefore a part of the recognition process. Thus the

ACORN method combines the convenience of automatic hierarchy selection with the efficiency which comes from using a predefined hierarchy in the recognition process.

In real-world applications, input is matched against several top-level templates. Current methods of hierarchical synthesis and interference matching involve matching the input against one template at a time. Such an approach is clearly undesirable for tasks such as speech recognition, which may involve large numbers of templates. The ACORN compiler takes a whole set of templates and produces a single, unified recognition network for it; common subtemplates are shared not just within top-level templates but also between them. An instance of a subtemplate is recognized just once -- not separately for each top-level template in which it occurs. Hence recognition time depends not on the total number of templates, but just on the number of templates which match some portion of the input. This property is encouraging, since the number of templates required to recognize a significant subset of English would probably be several thousand.

In sum, an ACORN can be looked at as a bottom-up version of an ATN; a parallel and non-deterministic version of a PARRY-like system; a general pattern-matcher; or an improved mechanism for hierarchical synthesis, with automatic hierarchy selection and subtemplate sharing between templates.

APPLICATIONS, IMPLICATIONS, AND EXTENSIONS

In order for an ACORN to be efficient, the templates and input data characteristic of the chosen problem domain should tend to be asymmetric, so that a template will usually match a given portion of the input in at most one way. Let us illustrate with a negative example. Suppose the template we wish to match is K_5(a, b, c, d, e), the complete graph on five vertices, represented by the conjunction of relations line(a, b) ∧ line(a, c) ∧ ... ∧ line(d, e). Then any occurrence of K_5 (as a subgraph, say) in the input corresponds to 5! = 120 instances of T, since there are 5! different ways to bind the variables a, b, c, d, e to the five vertices of the K_5 in the input. For symmetries on a larger scale, the problem grows combinatorially worse. Clearly, an ACORN would be inefficient in such a domain, since it would insist on finding all instances of every template.

Fortunately, many problem domains do not exhibit this bothersome property. Speech, in particular, is highly asymmetric, partly because it is embedded in a one-dimensional ordered temporal domain. If tell(t_1, t_2) is true, then $t_1 < t_2$, so tell(t_2, t_1) cannot be true. Symmetries at a higher level can occur only if there is more than one syntactically and semantically valid way to group the input words into phrases, i.e., if the input is inherently ambiguous.

What are the advantages of ACORNs for speech understanding? The bottom-up template-oriented approach is especially conducive to handling natural, idiomatic, conversational natural language robustly. Consider the problem in spoken speech of spurious insertions such as "oh," "um," "er." We wish to treat them the same as silences. We do this by adding rules like [oh(t_1, t_2) => SILENCE(t_1, t_2);] to our template grammar, and relaxing the test t_2=t_3 for temporal adjacency between two relation instances, such as tell(t_1, t_2) and me(t_3, t_4), to compute t_2=t_3 ∨ SILENCE(t_2, t_3).

This example also illustrates the reason for non-deterministic application of Colby's methods in a speech understanding system. Even if a spurious insertion is recognized, the corresponding portion of the input must not be discarded, since it may have been recognized incorrectly. If an ACORN recognizes an instance of "oh" in the interval [t_1, t_2], it puts the instance (t_1, t_2; ...) on the instance list of SILENCE, without discarding any information. That way, if the interval actually contains the word "no," it is still there for the lexical analyzer to find. In contrast, when PARRY ignores information, it throws it away altogether.

Another phenomenon common to conversational speech is the idiomatic expression, e.g., "How are you?" Using an ACORN, we can simply include explicit template rules for such expressions, e.g.,

[how•are•you(t_1, t_2) =>
GREETING(t_1, t_2); REPLY("Fine, how are you?")],

thereby short-circuiting the detailed syntactic parse which would be attempted by a more formal system such as Woods'.

The two techniques just described can be combined. Certain idioms such as "by the way" carry essentially no useful information and can be treated as spurious insertions by rules like

[by(t_1, t_2) ∧ the(t_2, t_3) ∧ way(t_3, t_4) =>
SILENCE(t_1, t_4);].

Some expressions occur either as meaningless idioms or as meaningful phrases, depending on context. Consider, for example, the utterance "I see, could I see the midnight digest?," which occurred in an actual experimental protocol. The first occurrence of "I see" is idiomatic and can be ignored; the second is essential to the meaning of the utterance. An ACORN, in processing this utterance, would recognize both occurrences as instances of SILENCE, without discarding any information. The first occurrence would be ignored, as desired, but the second one would still be available to match other templates.

Spurious deletions can also be handled by ACORNs. To handle spurious deletions, we want to permit partial matching of templates. We can do this within the ACORN framework simply by adding extra templates corresponding to commonly occurring partial matches of the original templates. The obvious weakness of this method is that it requires a priori knowledge of which deletions are likely to occur. The success of the method might require many iterations over a large corpus of test utterances, with new templates added as needed. Hopefully this process would converge, after a reasonable number of such iterations, to acceptable performance with respect to handling spurious deletions. (This method of "massive iteration" seems to have worked successfully for PARRY.)

Partial templates can be used for another purpose as well. Although the bottom-up approach has several advantages, as described above, it is useful to have certain properties associated with top-down processing. One such property is the ability to focus the attention of lower-level modules on critical portions of input. Another is the ability to hypothesize words from above, for lower-level modules to confirm or reject. Although we earlier referred to a lexical analyzer which finds all instances of primitive relations (words) in the input utterance, this would in practice be too expensive. The actual Hearsay II system seeks to constrain hypothesization as much as possible; to do this it applies high-level information to cut down the number of plausible words matched against each portion of the input. Thus it is desirable to have a speech understanding ACORN generate intermediate partial information telling the lower level modules which portions of the input they should concentrate on processing, and which words are likely to occur at a given place in the input, on the basis of the already recognized portions of the surrounding context.

This top-down extension to the basic bottom-up mechanism requires knowledge about the predictive value of partial templates. For example, we know that "What time" often occurs in the phrase "What time is it?" We can incorporate this information in an ACORN by including a rule

[what(t_1, t_2) ∧ time(t_2, t_3) =>
WHAT•TIME(t_1, t_3); TEST(t_3, t_{any}; "is it")],

where TEST is the action invoked upon recognition of the template. The effect of the TEST is to look for the missing instance of "is it" starting at the time t_3 in the input utterance. If it is found, it is added to the instance list for is-it, leading to the desired completion of the full template "What time is it."

In the above example, a partial template was used to predict downwards in the network. Partial templates can also be good upward predictors. For example, given an instance of the partial template T_1 = "time is it," the probability $P(T_2|T_1)$ that it occurs as part of the template T_2 = "What time is it" may approach certainty. If $P(T_2|T_1)$ is high enough, say .99, we may wish to save processing time by simply predicating that T_2 does in fact occur. Such a scheme is currently being implemented.

EVALUATION AND CONCLUSIONS

A full evaluation of the ACORN method must of course await experience with large-scale implementations. In the meantime, there are several properties we observe from the current, partial implementation.

(1) The recognizer is efficient.

(2) It is extremely easy to modify, since changes are restricted to the template grammar.

(3) Using an ACORN makes it possible to dispense with a formal parse.

(4) Even when an ACORN cannot fully parse an utterance, it can still provide a partial parse.

(5) ACORNs are organized so as to factor recognition processing into simple, universal, and independent operations performed at the nodes. This has made them trivial to implement and, in addition, makes them well-suited to parallel execution on a multiprocessor.

Finally, we expect ACORNs to have a broad range of applications, since they seem well-suited to recognizing any sort of relational pattern which manifests few symmetries. Both spoken utterances and real-world scenes appear to be in this class. At this point, we have implemented one ACORN processor for the syntax and semantics in speech (SASS) module of Hearsay II. Another ACORN processor has been built for recognizing the occurrence of inferred patterns (abstractions) in pattern learning training data. The abstractions themselves are produced by a program called Sprouter which grows a minimal ACORN to recognize all subtemplates common to two or more relational patterns. [5] From these experiences, it seems that ACORNs may provide an effective mechanism for general recognition.

REFERENCES

1. Barrow, H.G., Ambler, A.P., & Burstall, R.M. Some techniques for recognising structures in pictures. In S. Watanabe (Ed.), Frontiers of pattern recognition. New York: Academic Press, 1972.
2. Colby, K.M., Faught, B., & Parkison, R.C. Pattern-matching rules for the recognition of natural language dialogue expressions. Memo AIM-234. Stanford: Stanford Artificial Intelligence Laboratory, Stanford University, 1974.
3. Feldman, J.A., & Rovnar, F. An Algol-based associative language. Communications of the ACM, 1969, 12, 439-449.
4. Frost, M. The news service system. Operating Note SAILON 72-2. Stanford: Stanford Artificial Intelligence Laboratory, Stanford University, 1974.
5. Hayes-Roth, F. An optimal network representation and other mechanisms for the recognition of structured events. Proceedings of the Second International Joint Conference on Pattern Recognition, 1974.
6. Hayes-Roth, F. The representation of structured events and efficient procedures for their recognition. Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1974.
7. Hayes-Roth, F., and Mostow, D.J. An automatically compilable recognition network for structured patterns. Pittsburgh: Department of Computer Science, Carnegie-Mellon University, 1975.
8. Hewitt, C. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. Cambridge: MIT Project MAC, 1972.
9. Lesser, V. R., Fennel, R. D., Erman, L. D., & Reddy, D. R. Organization of the HEARSAY II speech understanding system. Proceedings IEEE Symposium on Speech Understanding, 1974.
10. Miller, P.L. A locally-organized parser for spoken input. Communications of the ACM, 1974, 11, 621-630.
11. Newell, A., Barnett, J., Forgie, J., Green, C., Klatt, D., Licklider, J.C.R., Munson, J., Reddy, R., & Woods, W. Speech understanding systems: final report of a study group. New York: American Elsevier, 1973.
12. Newell, A. Production systems: models of control structures. In W.C. Chase (Ed.), Visual information processing. New York: Academic Press, 1973.
13. Rulifson, J.F., Derksen, J.A., & Waldinger, R.J. QA4: a procedural calculus for intuitive reasoning. Menlo Park: Stanford Research Institute, 1972.
14. Woods, W.A. Transition network diagrams for natural language analysis. Communications of the ACM, 1970, 13, 591-606.

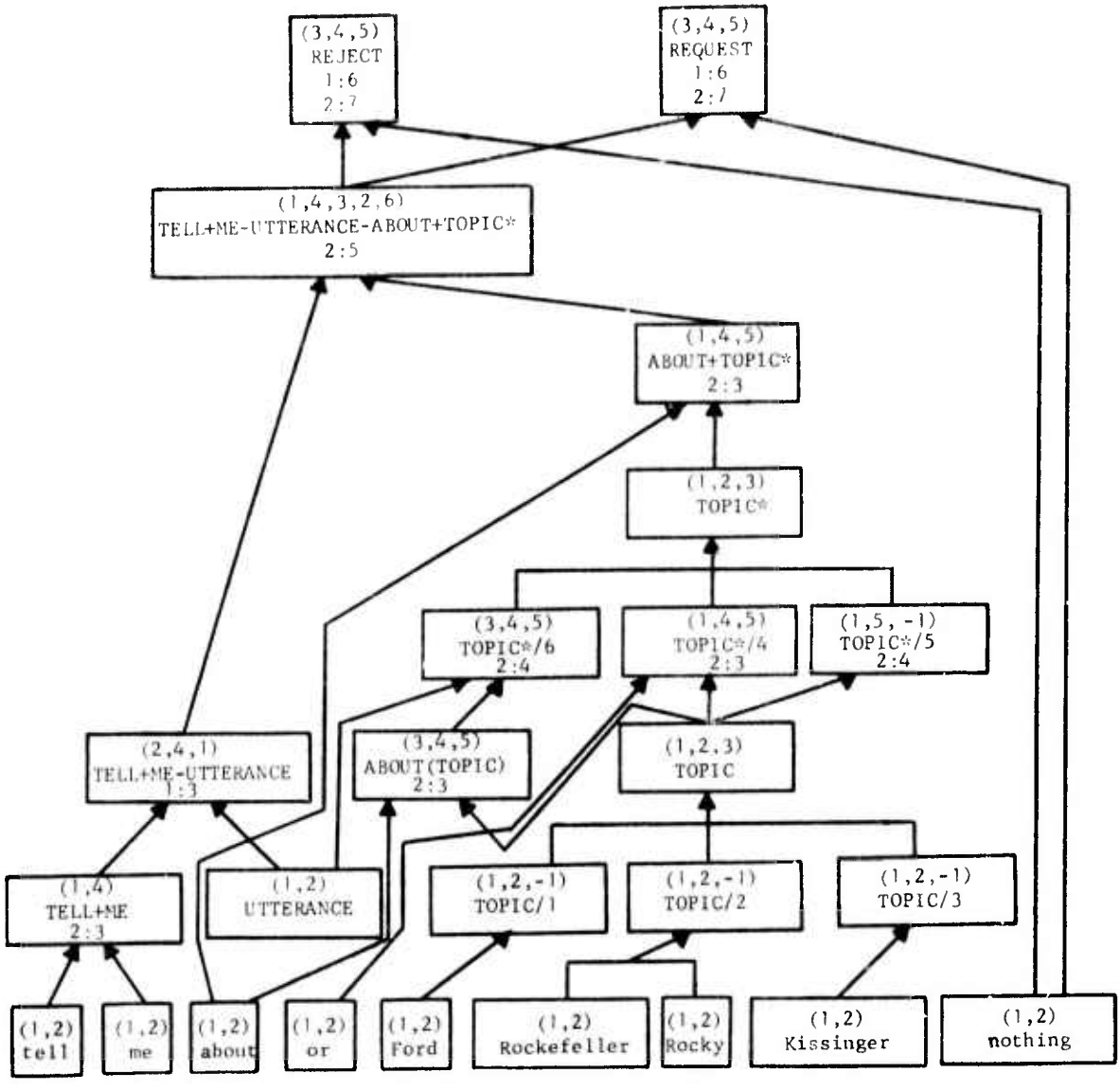


Figure 1.

Sample Recognition Network (an ACORN). See text for an explanation of the tests i:j below the nodes and the generators $(i_1, \ldots, i_k)$ above them.

SYNTAX AND SEMANTICS IN A DISTRIBUTED SPEECH UNDERSTANDING SYSTEM

Frederick Hayes-Roth and David J. Mostow
Computer Science Department[1]
Carnegie-Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

The Hearsay II speech understanding system being developed at Carnegie-Mellon University has an independent knowledge source module for each type of speech knowledge. Modules communicate by reading, writing, and modifying hypotheses about various constituents of the spoken utterance in a global data structure. The syntax and semantics module uses rules (productions) of four types: (1) recognition rules for generating a phrase hypothesis when its needed constituents have already been hypothesized; (2) prediction rules for inferring the likely presence of a word or phrase from previously recognized portions of the utterance; (3) respelling rules for hypothesizing the constituents of a predicted phrase; and (4) postdiction rules for supporting an existing hypothesis on the basis of additional confirming evidence. The rules are automatically generated from a declarative (i.e., non-procedural) description of the grammar and semantics, and are embedded in a parallel recognition network for efficient retrieval of applicable rules. The current grammar uses a 450-word vocabulary and accepts simple English queries for an information retrieval system.

INTRODUCTION: THE PROBLEM

The fundamental problem facing the syntax and semantics component of a speech understanding system is uncertainty. The system is uncertain about a variety of questions, including: whether a given word is really uttered by the speaker; when a recognized word begins and ends; whether a particular interval of the utterance contains a silence, a filled pause ("er," "um," "uh"), an informationless interjection ("y'know," "I mean"), or an information-bearing word or phrase; whether a recognized word or phrase is used in a particular sense; etc. Any decisions made on the basis of such uncertain information are potentially incorrect and must therefore be reversible. The classical method of reversing decisions is backtracking. Backtracking and best-first evaluation of alternative parses are the primary strategies employed by the Hearsay I speech understanding system (Reddy, et al., 1973a, 1973b).

In Hearsay II (Lesser, et al., 1975) multiple alternatives are represented explicitly in a global data structure ("blackboard") and considered in parallel rather than one at a time as in Hearsay I. Processing is driven by independent data-directed knowledge source modules (KSs) which create, examine, and revise hypotheses, stored on the blackboard, about the utterance. One dimension of the blackboard is level of representation: an interval of speech may be simultaneously represented at the acoustic, phonetic, phonemic, syllabic, word, phrasal, and conceptual levels. The KSs translate from one level to another with the ultimate objective of representing the utterance at the conceptual level, i.e., understanding it. Hearsay II is a distributed logic system in that control of processing is distributed heterarchically among the KSs rather than organized hierarchically. Each KS is responsible for deciding when it has useful information to contribute to the analysis of the input.

The syntax and semantics KS in Hearsay II is called SASS, and deals with hypotheses representing words and phrases perceived or expected in the utterance. From SASS's viewpoint, the blackboard can be viewed as a chart of hypothesized words as in Figure 1, which represents the word hypotheses generated by lower-level KSs in response to the utterance "Tell me about beef." In the figure, time goes from left to right and the vertical dimension represents hypothesis credibility on a scale from -100 to 100, as estimated by other KSs. SASS's problem is to find the most plausible sequence of temporally adjacent words. Plausibility is defined by the credibility of the individual word hypotheses and the grammaticality and meaningfulness of the sequence. The concept of temporal adjacency is generalized to tolerate fuzzy word boundaries, overlap between successive words, silences in the middle of word sequences, and unintelligible intervals. Since some of the uttered words may not have been hypothesized, SASS must be able to expand the solution space by inferring the likely presence of a missing word on the basis of existing word hypotheses. Such inferences are relatively weak since several predictions may be plausible in a given context. In the example of Figure 1, SASS hypothesizes the missing word "tell" in the interval preceding "me about beef." Since SASS is uncertain as to which word hypotheses are correct, it also makes several incorrect word predictions. Figure 2 shows the words predicted by SASS on the basis of the words shown in Figure 1. The figures do not reflect the fact that the various hypotheses are generated at different times and SASS starts generating predictions prior to completion of the word recognition process.

In order to control the potentially explosive search through this combinatorial and expanding solution space, SASS must be able to reflect the variable reliability of its inference rules and to relax its plausibility criteria dynamically so as to stimulate processing on unrecognized portions of the utterance. SASS must be able to use partial information to guide further processing in useful directions. To avoid duplicated computation, SASS must store and use partial parses, which are intermediate computations (plausible subsequences) common to many potential parses. SASS must combine these partial parses into plausible complete parses, select the best complete parse, interpret the meaning of the recognized utterance, and respond appropriately.

The problems faced by SASS -- uncertainty, combinatorial search, fuzzy pattern-matching, strong and weak inferences, and the need to exploit partial information -- are common to many large knowledge-based systems. Efficient solution of these problems appears to require a system organization in which the scheduling of inferential processes is sensitive to various cooperative and competitive relationships among the inferred hypotheses. For example, processing should be facilitated on an hypothesis supported cooperatively by multiple sources of information. Conversely, processing should be inhibited on an hypothesis which competes -- i.e., is inconsistent with -- a strongly credible hypothesis. Inhibition in an environment of uncertainty must be implemented non-deterministically, since the weaker hypothesis may in fact be correct. Non-deterministic inhibition is effected in Hearsay II by a focus of attention mechanism which allocates computational resources so as to consider the most promising hypotheses before others (Hayes-Roth & Lesser, 1976).

The approach used in SASS is relevant to pattern recognition for its fuzzy pattern-matching; to problem solving for its flexible combination of bottom-up, top-down, forward inferencing, and problem reduction mechanisms; and to information retrieval and the problem of pattern-directed function invocation for its efficient mechanism for continuously monitoring a data base for occurrences of any of a large number of relational patterns or templates.

1 This research was supported in part by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

OVERVIEW OF METHOD

Given a declarative (i.e., non-procedural) description of the target language which our system is to understand, we need to convert it into behavior which is adequate to understand utterances in the language efficiently and robustly. Our approach has been to automate this conversion as much as possible. Syntactic and semantic knowledge about the target language is expressed in a compact, readable grammar. A compiler converts the grammar into precondition-response productions. The productions are embedded in a recognition network to enable efficient continuous monitoring of the blackboard for stimuli matching production preconditions. In general, many productions will be invocable at any given time. Various scheduling policies serve to hasten the invocation of productions which are considered likely to generate useful (correct, relevant, and necessary) results and to inhibit or defer less promising invocations.

LINGUISTIC KNOWLEDGE

The grammar describing the target language is expressed using parameterized structural representations (PSRs), which are sets of attribute-object pairs. We use a PSR to define a class of words and phrases which can fulfill the same syntactic or semantic function in the target language. The current target language consists of simple English queries for a news retrieval program. For example, the PSR

```
($CLASS: $QUERY, $PNAME: "PARSED QUERY",
        <: $GIMME+$WHAT,
        <: TELL+$ME+$RE+$TOPICS,
        <: WHAT+HAPPENED+$ANYWAY,
        <: WHAT+$BE+THE+$NEWS+$RE+$TOPICS,
        <: $BE+THERE+$ANY+$PIECES+$RE+$TOPICS,
        $ACTION: PASS,
        $LEVEL: 300)
```

defines the class "$QUERY" of possible queries in terms of its alternative syntactic realizations. The attribute "<" denotes membership in the class. Each member of the class is a sequence template whose constituents, separated by "+", are words or phrases. Phrasal constituents are prefixed by "$" and defined in turn by other PSRs. Additional attributes of the class are defined by other components of the PSR. "$ACTION: PASS" means that SASS's response upon recognizing an instance of any of the five templates in the class should be to treat it as an instance of $QUERY. The $LEVEL attribute estimates the relative completeness of the partial parse underlying the hypothesized phrase. The PSR

```
($CLASS: $TOPICS,
        <: $PLACE,
        <: $FOOD,
        <: $TECHNOLOGY,
        <: $SCIENCE,
        <: $GOVERNMENT,
        <: $POLITICS,
        <: $PEOPLE,
        <: $TOPICS+$CONJUNCTION+$TOPICS,
        $ACTION: PASS, $LEVEL: 40)
```

defines the class of possible topics in the news in terms of its semantic subclasses. The grammar for the current 450-word target language consists of 113 PSRs.

TYPES OF BEHAVIOR RULES

SASS has a repertoire of strong and weak methods, represented by different types of behavior rules used in understanding.

A recognition rule generates a phrase hypothesis in response to sufficiently credible hypotheses for the phrase's constituents. SASS considers an hypothesized constituent to be recognizable if its credibility rating, determined by other KSs, exceeds a minimum threshold for plausibility. The hypothesized constituents may also have to satisfy some structural condition such as temporal adjacency between sequential constituents of a phrase. A recognition rule represents a strong inference; its strength is the probability that the recognized constituents can be interpreted as an instance of the phrase. For example, "beet" can be interpreted as a food or as a complaint, depending on context. Recognition rules drive processing upward toward a complete parse of the utterance from plausible partial parses. Recognition behavior can be thought of as bottom-up parsing.

A prediction rule hypothesizes a word or phrase which is likely to occur in the context of a previously recognized portion of the utterance. Prediction rules drive processing outward in time from "islands of plausibility," and are necessary since not all words in a spoken utterance may be recognized bottom-up by lower-level KSs. Predictive behavior can be thought of as forward inferencing. The strength of a predictive inference is the conditional probability that the predicted constituent occurs, given that its predictive context has been recognized. This strength is inversely related to the number of constituents which can plausibly occur in the given context.

A respelling rule enumeratively hypothesizes the constituents of a predicted phrase, by subdividing an hypothesized sequence into hypotheses for its sequential constituents, or by splitting an hypothesized class into alternate hypotheses for its various members. Respelling rules drive processing downward toward the word level, so that high-level phrasal predictions can ultimately be tested word-by-word by lower-level KSs. Respelling can be thought of as top-down behavior or generation of subgoals from goals.

Finally, a postdiction rule solicits post hoc support for (i.e., serves to increase the credibility ratings of) existing hypotheses from other hypotheses in whose context they are plausible. Postdiction rules include prediction and respelling rules which are too weak to justify creation of hypotheses, but can contribute useful information when the hypotheses already exist. For example, an expectation for an instance of $TOPICS following the word "about" should not be respelled into hypotheses for all the nouns in the vocabulary, since to do so would explode the search space. However, once the word "beet" is hypothesized in the correct time interval on the basis of other knowledge, the hypothesis should receive support from the expectation for a topic word.

Postdiction rules serve three functions: they allow cooperation between inferences which support the same hypothesis on the basis of different evidence; they allow words and phrases hypothesized with initial low credibility ratings to be recognized on the basis of their contextual plausibility; and they help focus attention in productive directions by increasing the ratings of hypotheses which are contextually plausible (and thus relatively likely to be correct) so that processing on them is scheduled sooner. In the sense that postdiction responds to weakly-rated hypotheses by seeking causal antecedents (predictors) for them, postdiction can be thought of as post hoc inferencing or "twenty-twenty hindsight."

CONVERSION OF STATIC KNOWLEDGE TO BEHAVIOR RULES

Most of the information necessary for understanding the target language is implicit in the grammar which describes it. The automatic conversion of this static information into a usable procedural form is effected by a simple compiler called CVSNET, which translates the PSRs into recognition, prediction, respelling, and postdiction rules. A few rules hand-coded in explicitly procedural form are then added, for example a rule that prints a message when a sentence is recognized. The only linguistic knowledge in CVSNET itself is an elementary understanding of sequences and classes. CVSNET decomposes the sequence templates $c_1+c_2+...+c_n$ into pairs of subsequence templates. For example, from the sequence template TELL+$ME+$RE+$TOPICS, CVSNET generates the new templates $ME+$RE+$TOPICS and $RE+$TOPICS.

CVSNET then generates the appropriate rules for each template. The recognition rule for a sequence is to concatenate its hypothesized subsequences provided they are temporally adjacent and sufficiently credible. The respelling rule respells a predicted sequence into its two subsequences. Prediction rules

are generated to predict the remaining constituents of the sequence when a subsequence of it has been recognized. Similarly, CVSNET generates rules for recognizing an instance of a class from an hypothesized constituent of the class and for respelling a predicted class into its constituents. CVSNET estimates the strength of each such rule as an inverse function of class size. CVSNET also generates the relevant postdiction rules. Some of the rules generated from the PSRs are shown below; rule type is indicated by the type of arrow separating stimulus and response ("->" for recognition, "=>" for prediction, "+>" for respelling, and "<=" for postdiction) and rule strength is shown in parentheses.

TELL & $ME -> TELL+$ME < CONCATENATE (100) (100) >

TELL & $ME <= TELL+$ME < POSTDICT!SEQ (100) (100) >

TELL+$ME +> TELL & $ME < RESPELL!SEQ (100) (100) >

$ME => TELL < PREDICT!LEFT (50) >

TELL <= $ME < POSTDICT!LEFT (50) >

TELL => $ME+$RE+$TOPICS < PREDICT!RIGHT (100) >

$ME+$RE+$TOPICS <= TELL < POSTDICT!RIGHT (100) >

$FOOD -> $TOPICS < PASS (100) >

$TOPICS +> $FOOD < RESPELL!CLASS (70) >

$FOOD <= $TOPICS < POSTDICT!ELEMENT (88) >

The linguistic knowledge expressed compactly in the grammar is represented highly redundantly in the generated rules. This redundancy provides the basis for robust performance in the errorful domain of speech: in regions of the utterance where strong inferences (recognition rules) are inadequate (for example, because lower-level KSs have failed to hypothesize some of the uttered words), weaker inferences must be applied in order for the utterance to be understood.

IDENTIFICATION OF INVOCABLE RULES

All of the rules described have the form $[\text{precondition}(x_1,x_2,\ldots,x_n) \Rightarrow^f \text{response}(x_1,x_2,\ldots,x_n)]$, signifying that a specified response can be inferred with strength f from the objects x_1, x_2, ..., x_n whenever these objects are in the relationships described by the associated precondition. The large number of rules required even in a relatively simple system (over 3000 rules for a 450-word vocabulary) necessitates an efficient means of continuously monitoring the blackboard to determine which rules are currently invocable because of data satisfying their preconditions.

This problem is solved by embedding the rules in an automatically compilable recognition network (ACORN), as discussed elsewhere (Hayes-Roth & Mostow, 1975). In brief, each grammatical constituent (word or phrase) is assigned a unique node in the network. Rules whose preconditions refer to the constituent are stored at the node. Whenever an hypothesis for the constituent is created or revised, its node is activated and the relevant rules become invocable.

PRINCIPLES OF CONTROL

The rule preconditions are defined in terms of various thresholds for plausibility, temporal adjacency, etc. These thresholds can be given values specific to a particular region of the utterance and are dynamically modifiable. Thus rules are invoked not only in response to new hypotheses but also in response to local threshold changes. This mechanism allows flexible matching of rule preconditions. Thresholds can be relaxed in unrecognized regions of the utterance to permit localized application of methods whose weakness would cause combinatorial explosion if they were applied uniformly throughout the utterance.

Hypotheses are explicitly linked in the data base to hypotheses which support them inferentially, and the links are marked with the strengths of the inferences. A rating policy module (RPOL) rates the plausibility of new hypotheses on the basis of the ratings of the hypotheses which support them and the strengths with which they do so. RPOL updates these ratings when an hypothesis receives new support or when the rating of one of its supporting hypotheses is changed. Hypotheses are rated separately on their contextual plausibility and on the extent to which they are supported by lower-level hypotheses.

The combinatorial search can be controlled by modifying the appropriate threshold values. For example, the search can be broadened or narrowed by relaxing or tightening criteria for recognizability, since the solution space consists only of sequences of recognizable words. A best-first search policy can be implemented simply by ordering rule invocations according to the strengths of the rules and the plausibility ratings of the hypotheses matching the rules' preconditions. The search can be further focussed by inhibiting low-level processing within a region already accounted for by a credible high-level hypothesis. Of course this policy must be pursued with caution since the high-level hypothesis may be incorrect. Cautious inhibition is implemented as deferred processing. A similar policy of procrastination can be used to deter application of weak inferences in a region until strong methods fail. An inferential process can be deterred by scheduling it with low priority (so that it may never in fact be executed), or by scheduling it only when the relevant thresholds are relaxed. The latter mechanism permits reconsideration of previously rejected alternatives.

Discourse rules can also help to focus the search. For example, an hypothesis that the current topic of conversation is food increases the a priori probability that the word "beef" will be uttered. If we can predict subject matter or syntax from any one of many knowledge elements (e.g., a recognized cue word in the same utterance, semantic analysis of previous utterances, knowledge of the particular speaker's interests), we can create such an hypothesis. This form of semantic and syntactic priming is non-restrictive in that it does not preclude recognizing an utterance which is inconsistent with an hypothesized topic of conversation or an expectation for a particular grammatical construction. The mechanism is also graceful in that it does not impose a strict hierarchy of topical domains, and in fact tolerates ambiguity and uncertainty in the expectations generated by previous discourse.

Inexact matching can also be carefully controlled with thresholds. An interval of silence in the middle of an utterance can be accepted by relaxing temporal adjacency thresholds in the region of the silence so that hypothesized sequence constituents temporally separated by the silence will be considered temporally adjacent. For example, if the speaker says "Tell me about . . . beef," this mechanism allows the words "about" and "beef" to be considered temporally adjacent. Interjections and unclear intervals of speech can be nondeterministically ignored by treating them as silences. Sometimes the uttered words cannot be recognized by lower-level KSs even after SASS hypothesizes them on the basis of surrounding context. In such cases, partially-matched phrases can be recognized by lowering credibility thresholds in unintelligible intervals so that unfulfilled expectations for missing constituents are treated as if they had been fulfilled. These mechanisms can even be used to tolerate some variation from the target language by ignoring extra verbiage not accounted for in the grammar and by filling in omitted constituents required by the grammar.

PERFORMANCE EVALUATION

The contribution of each KS in Hearsay II is highly dependent upon the behavior of the others. Consequently, SASS's performance is difficult to evaluate. For instance, SASS's prediction of the missing word "tell" in the previous example may have been critical to recognition of the utterance. On the other

hand, the word-hypothesizer KS might eventually have lowered its own thresholds enough to have weakly hypothesized the missing "tell." In this case, SASS's postdiction of the hypothesized "tell" from its surrounding context might have been critical in increasing its credibility rating sufficiently to permit it to be recognized.

Despite the complex dynamics of the integrated system, we do have an evaluation methodology for SASS which will be pursued in the next year. Basically, our strategy is to generate a variety of artificial problems, each defined by a set of hypothesized words, and measure the elapsed time until SASS parses the utterance. In particular, we should be able to evaluate the relative efficacy of the four types of behavior rules in overcoming various kinds of error in the artificial input. If we can then estimate the relative frequencies of different kinds of errors generated by lower-level KSs, we can attempt to optimize SASS's behavioral profile.

CONCLUSION

There are many functions to be performed by a syntax and semantics knowledge source within a speech understanding system. In addition to simply parsing a sentence, the knowledge source must use a variety of strong and weak interencing methods to hypothesize missing constituents and adduce support for existing hypotheses found in appropriate contexts. A production system using four types of rules has been developed to implement such desirable "knowledgeable" behaviors, which are automatically interred from a simple declarative representation of the language to be understood. By making the invocation of a rule be dependent upon both the credibility of the data matching the rule's preconditions and the estimated strength of the rule as a useful inference, the entire search process may be controlled so as to pursue dynamically modifiable global and local processing objectives. In sum, such a production system provides a general framework for representing "knowledgeable" syntactic and semantic behaviors. Moreover, the fine computational grain of the behavior rules makes possible the flexible and precise control needed to avoid a combinatorial explosion in the search for a plausible interpretation of continuous speech.

REFERENCES

Hayes-Roth, F. and Lesser, V. R. Focus of attention in a distributed logic speech understanding system, 1976. Appears in this volume.

Hayes-Roth, F. and Mostow, D. J. An automatically compilable recognition network for structured patterns. Proc. Fourth Inter. Joint Conf. Artificial Intelligence, 1975, 246-251.

Lesser, V. R., Fennell, R. D., Erman, L. D., and Reddy, D. R. Organization of the HEARSAY II speech understanding system. IEEE Trans. Acoustics, Speech, and Signal Processing, 1975, ASSP-23(1), 11-23.

Reddy, D. R., Erman, L. D., and Neely, R. B. A model and a system for machine recognition of speech. IEEE Trans. Audio and Electroacoustics, 1973a, AU-21(3), 229-238.

Reddy, D. R., Erman, L. D., Fennell, R. D., and Neely, R. B. The HEARSAY speech understanding system: an example of the recognition process. Proc. Third Inter. Joint Conf. Artificial Intelligence, 1973b, 185-193.

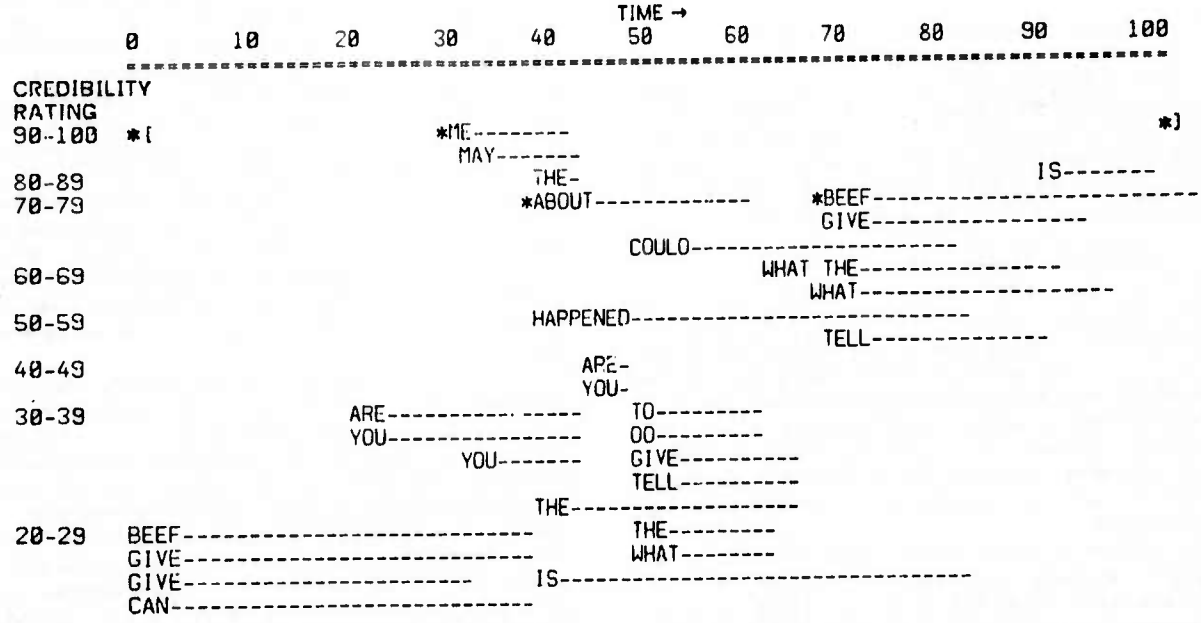


Figure 1. Words hypothesized bottom-up in response to utterance "Tell me about beef" "*" marks correct hypothesis; "[" and "]" denote hypothesized beginning and end of utterance

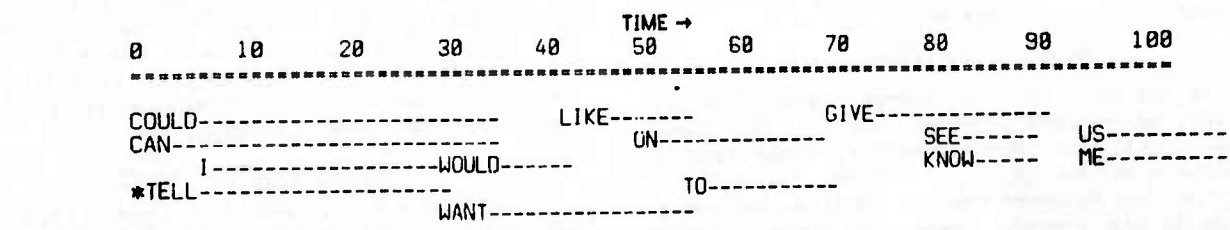


Figure 2. Words predicted by SASS on the basis of the hypotheses shown in Figure 1

DISCOURSE ANALYSIS AND TASK PERFORMANCE IN THE HEARSAY II SPEECH UNDERSTANDING SYSTEM

Frederick Hayes-Roth, Gregory Gill, and David J. Mostow
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania
January 28, 1976

ABSTRACT

While much of the syntactic and semantic analysis of a spoken utterance in HEARSAY II is performed by the syntax and semantics knowledge source module (SASS), the final semantic interpretation is produced by a discourse analysis module (DISCO). Using knowledge about the state of the conversation, DISCO interprets the speaker's intention and directs the appropriate activities within the task program. In most cases, the intention of the speaker is to establish a general topic of interest, to retrieve articles by keyword expressions, or to have retrieved articles written on an output device. In some circumstances, DISCO can predict the likely subsequent occurrence of a particular type of phrase. Such predictions cause SASS, at the outset of analyzing a new utterance, to create corresponding phrasal goals which effect a semantic bias (subselection, perceptual set) that influences which words are hypothesized and how plausible they appear.

As described elsewhere [Hayes-Roth and Mostow, 1976], a spoken utterance is recognized if the syntax and semantics knowledge source module (SASS) can generate a grammatical parse. In the current task (document retrieval by keyword expression), there are seven distinct types of intentions reflected by SASS's grammar and distinguished by the discourse analysis module (DISCO). The most frequent is the request, a command to retrieve documents by keyword expression (e.g., "Give me news about Ford"). The next most frequent is a selection, a statement which identifies general semantic areas or attributes of documents of interest (e.g., "Select from stories about politicians"). These two functions -- formulating a keyword retrieval expression and selecting a semantic area -- can be performed simultaneously by sentences of type request-and-select (e.g., "Give me news on Ford, the politician"). The grammar contains the information that some words (e.g., "politicians" or "politician") are cues for menus of keyword expressions (e.g., the menu $POLITICIAN includes "Ford", "Rockefeller", etc.). Recognition of such a cue causes generation of goals for finding words or phrases from the associated menu. If strong enough, these goals are respelled (enumerated) into their constituent keyword expressions by SASS. The word hypothesizer (POMOW) is sensitive to such goals and will attempt to generate supporting word candidates in the appropriate time region.

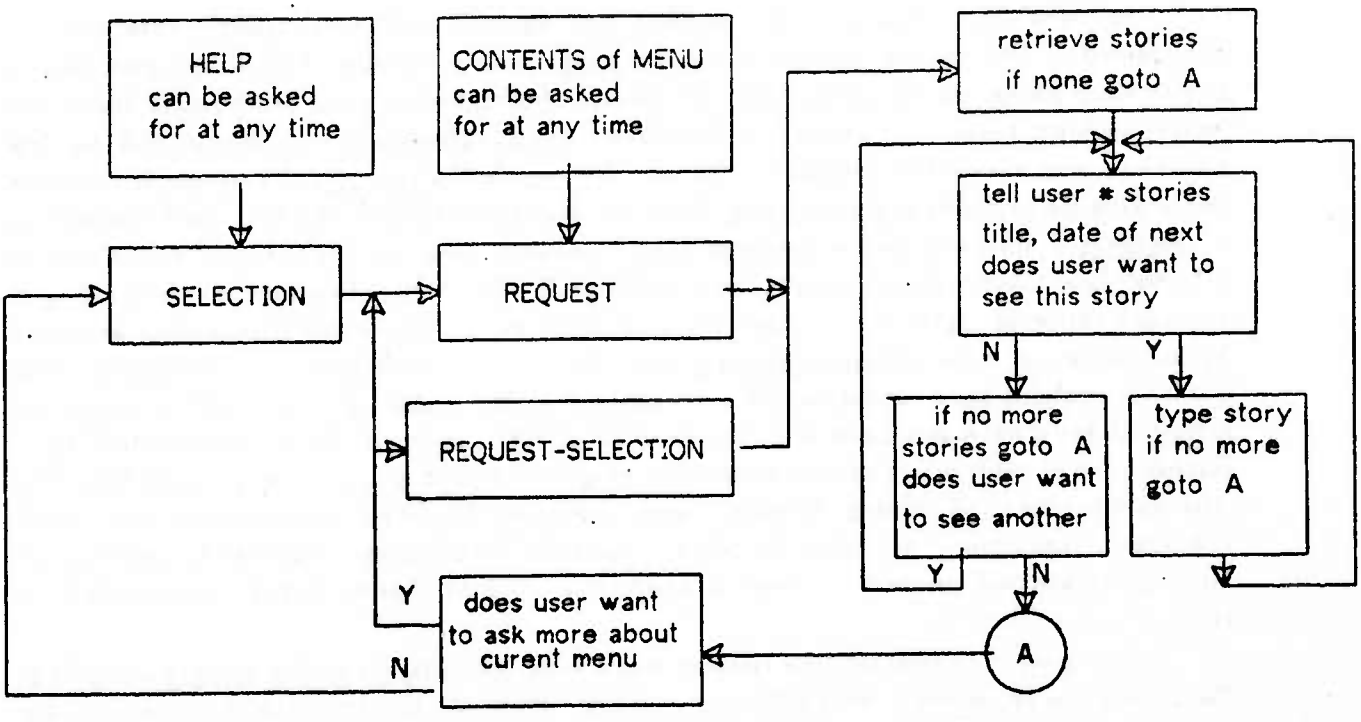
Once a menu is selected, the speaker may ask for the contents of the current menu (e.g., "What are the keywords") and DISCO will enumerate them. At any time, the user may ask for help (e.g., "Help" or "What are the topics?") and DISCO will describe the various content areas. Finally, DISCO will sometimes ask questions (e.g., "Do you want to see other stories on Ford?") with an expectation for yes or no type responses.

The typical conversational flow is repesented by a probabilistic finite state automaton which, along with a list of menus, cues, and contents, is input to DISCO at initialization. Each

time SASS parses an utterance, its type and semantic content are passed to DISCO. The appropriate response is determined and the state of the conversational model is updated. DISCO performs whatever action is indicated, such as reporting the number of articles matching the keyword expression, printing an article, supplying helpful information, or asking a clarifying question. DISCO passes to SASS its predictions about which type of utterance will follow and which semantic classes are likely to be mentioned. SASS establishes these predictions as phrasal goals on the blackboard. During processing of the subsequent utterance, SASS and POMOW will exhibit a bias toward hypothesizing words and phrases which are consistent with the phrasal goals. Such semantic subselection of words predicted top-down can facilitate recognition of content expressions in contexts where many different words are syntactically permissible. If the subsequent utterance violates the expectations of the conversational model, successful recognition will not be precluded but will probably be somewhat slowed because of unfavorable scheduling competition from knowledge source activities attempting to realize those expectations.

REFERENCE

Hayes-Roth, F., and Mostow, D. J. Syntax and semantics in a distributed speech understanding system. Proceedings of the 1976 I.E.E.E. International Conference on Acoustics, Speech and Signal Processing, (in press). Also appears in this volume.



Finite state conversational model for news retrieval task

Segmentation and Labeling of Connected Speech

Henry G. Goldberg
Computer Science Department†
Carnegie-Mellon University
Pittsburgh, Pa. 15213

January, 1976

Introduction

The development of a parametric level knowledge source for Hearsay II has been closely tied to other research reported by Goldberg [GOL75]. In that work, a number of design choices for the initial signal-to-symbol transformation were investigated within the framework of straightforward segmentation and labeling algorithms. The segmentation and labeling programs in that study were developed to be relatively independent of such a design dimension as input parametric representation. They rely rather heavily upon empirical knowledge about the parametric (acoustic) nature of phonetic phenomena, and upon some basic methods of statistical pattern classification. Good performance was achieved with these algorithms, and some of the design choices were shown to be irrelevent to the goal of achieving accurate machine transcription of connected speech at the acoustic-segmental (sub-phonemic) level of representation. This working paper consists of a brief summary of the comparative performance evaluations reported by Goldberg [GOL75] and of the particular configuration currently being used as a front end transcriber for Hearsay II.

Background

Although most of our knowledge about how to recognize and understand speech is taken from human performance, the structure of computer speech understanding systems and speech recognition schemes is of particular importance to this study. Knowledge

† This research was supported in part by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

about speech is generally organized into separate sources of knowledge, which work with a representation of the information content of the input utterance. These representations may exist at a number of different levels, as suggested by their elements: speech sounds, phonetic gestures, phonemes, syllabels, words, syntactic units, concepts, etc. In evaluating the performance of recognition processes at the parametric representation level, we eliminate, as much as possible, the effects of ambiguities from other levels. Such ambiguities as coarticulation or phonological variation will strongly affect the degree to which the expected transcription of an utterance corresponds with the acoustic performance. A great difficulty in comparing published results, is that the level of the knowledge used in recognition and the representation used for evaluation is not usually specified. Usually, only total system performance may be compared, not the effectiveness of component methods.

Parametric Representations

Parametric representations fall into a few major types, and typical examples of each have been chosen for study. A bank of broad bandwidth filters (ZCC) with amplitude and zero-crossing measurements, and a bank of narrow band filters (ASA), amplitude only, represent analog methods. A digital Fourier transform of the LPC filter [MAR72] produces a smoothed spectral envelope (SPG) very much in current use. Finally, the autocorrelation sequence (ACS) is employed with a special method designed for it.[ITA75] Each method yields a set of measurements at uniform, short intervals -- a pattern.

Distance Metrics

Distance functions, chosen from Pattern Classification theory, are then applied to the parameter patterns as measures of acoustic similarity. The basic model adopted from that theory is that of a vector of parametric measurements for each pattern. These vectors define a space of possible patterns, within which a measure of distance may be applied between patterns. As populations of sample patterns are accumulated, better statistical descriptions may be estimated of the true distribution of those patterns in the space. A

simple example might be to collect all the occurences of a phone, and compute the mean and variance of each dimension. Then a suitable measure of similarity might be Euclidean distance, weighted by variance, to approximate a measure of the deviation from population mean. This is one distance metric chosen (SIG). The others are Euclidean distance (EUC), Correlation (COR) -- the magnitude normalized dot product, and Maximum Likelihood (LIK). In this last, the population covariance matrix is used to calculate Pr{unknown produced from population}, under the assumption of Gaussian distributions.

Segmentation

A method for segmenting speech into isolated, acoustically consistent segments is presented. The method is fairly independent of the choice of parametric representation, since it relies upon the acoustic similarity measure as the primary evidence of acoustic change. First, however, a threshold is applied to the signal amplitude measurement to detect speech/silence. Then the speech portion is examined further. In collecting evidence for a segment boundary, a measure of change is applied to neighboring parameter patterns. This produces a time sequence of values, whose peaks are detected and subjected to a threshold for acceptance or rejection. A composite of such functions yields the final segmentation. Narrow and broad pattern similarity, as well as amplitude change are the three functions applied during speech portions of the signal. This process is very much like the process hypothesized in the basic model for Signal Detection [EGA64]. That model may be applied to the problem of evaluating segment boundary "detectability."

Missing and extra segment errors are found to be as good as 4% and 19%, respectively. Significant differences in the segmentation effectiveness of the parametric representations is found. They may be ordered as follows: SPG, ACS, ASA, and ZCC. The best performance is found to be comparable to the state of the art. Little reduction in accuracy is encountered when new speakers are tested.

Table 1 shows the results of segmentation for 40 sentences from the News Retrieval task, one speaker. The reference segmentation contains 1082 segments primarily at the

phonemic level of description. The second reference contains corrections to this file, to make it more an acoustic description of the corpus. It has 1541 segments. The number of machine segments reported may be greater than the sum of this number (hand reported acoustic segments) and the number of extra boundaries. The discrepancy is merely the result of the way we evaluate segmentation by boundaries. Occasionally, two machine boundaries will fall close enough to a hand boundary to both be accepted. Such segments must, therefore, be very short, and are usually transition segments which may be easily detected at higher levels. The number of missing boundaries (segments), divided by the number of boundaries which are included in both reference segmentations, is the missing segment error rate. The number of shifted boundaries is also divided by this number. The number of extra boundaries is divided by the number of primary segments (the size of H1 in this case). The extra segment rates in parentheses are those where division is by the number of acoustic segments (size of H2). The value, d', is a single measure of detectability from the Signal Detection model. It has the effect of normalizing for the trade-off between missing and extra segment errors.

	SPG	ACS	ASA	ZCC
Segments #				
H1	1082	1082	1082	1082
H2	1541	1541	1541	1541
M	2026	2082	2010	2290
Missing				
#	37	57	91	52
%	3.7	5.8	9.2	5.3
Extra				
#	299	391	434	681
%	27.6	36.1	40.0	63.0
	(19.4)	(25.3)	(28.1)	(44.2)
Shifted				
#	28	34	45	41
%	2.8	3.4	4.5	4.1
d'	2.38	1.93	1.58	1.29
	(2.65)	(2.24)	(1.91)	(1.77)

Table 1: Segmentation -- Parametric Representation Dimension

Labeling

Labeling is accomplished by the same pattern similarity distances metrics. Given a set of phonetic elements as the recognition targets, a set of templates for each target is collected from the training data. This is achieved by a clustering algorithm developed with the purpose in mind of encoding some of the ambiguities encountered in phone performance into the set of templates. The pairwise distances are computed for all pairs of sample patterns in the training population for a particular phonetic target. Then a threshold is chosen from these values, and the distances below threshold are marked. The sample pattern in the most marked pairs is chosen as a representative template and all its marked mates are discarded. After iterating, the population is divided into clusters of various sizes, each with a "best" representative template pattern. Clusters of sufficiently small size are ignored.

Labeling itself proceeds by computing the distance from the unknown pattern to each template. In addition to the distance metrics mentioned, three prosodic features of each segment -- the average amplitude, the duration, and the amplitude contour of the surrounding segments -- are used to increase the distances to templates whose prosodic features were considerably different.

The set of templates (and their appropriate target labels) and the distance scores give the total recognition information available from this straightforward labeler. If some criterion is placed on the templates which one is willing to report to the rest of a system, then accuracy may be measured as a function of the severity or looseness of that criterion. If the true effect, upon a speech recognition system, of loosening the acceptance criterion is to be understood, one must also measure the expected number of separate targets reported at each instance. We call this the Branching Factor, and collect it as well as accuracy statistics in evaluating labeling performance.

Little difference is observed along the parametric representation or the classification metric dimensions, except for poorer performance for ZCC input. Each input segment is labeled as one of a set of 40 phone labels. The correct phone appears as the

first choice 28% of the time. It appears in the first three choices 55% of the time. However, when a lower level, acoustic transcription is used as evaluation referent, these values increase to 42% and 65%. Even the 28% accuracy, which arises from a comparison against phonemic expectation, is acceptable performance. It is the same as or slightly better than human spectrogram reading performance in the absence of other linguistic clues.[SH074]

Table 2 shows overall labeling accuracies for the four parametric representations.

p	SPG	ASA	ZCC	ACS
1	24.6(1.0)	27.1(1.0)	20.3(1.0)	28.7(1.0)
2	42.4(1.9)	39.1(1.9)	31.4(1.9)	44.4(1.9)
3	54.0(2.8)	50.4(2.8)	42.0(2.8)	54.6(2.7)

Table 2: Parametric Representation Dimension

The distance metric is the Euclidean distance function†, and a set of 40 phonetic recognition targets is used. The values reported for position p are Pr{correct template in position ≤ p}. The expected number of different targets (branching factor) is given in parenthesis.

Figure 1 is a graphic display of accuracy versus Branching Factor for the SPG/SIG experiment. Five plots are given, identified by the size of the target set used in each evaluation. The BF plot gives a particularly convenient view of accuracy against the demands that will be made upon higher levels by excess options in recognition.

Contributions

The major contributions of this research are in three areas. First, a clear picture is provided of the segmentation and labeling performances available from standard parametric representations. An ordering can be made of the representations, for segmentation effectiveness, which agrees fairly well with existing beliefs about the

† The ACS representation was run only with Itakura's log ratio distance.

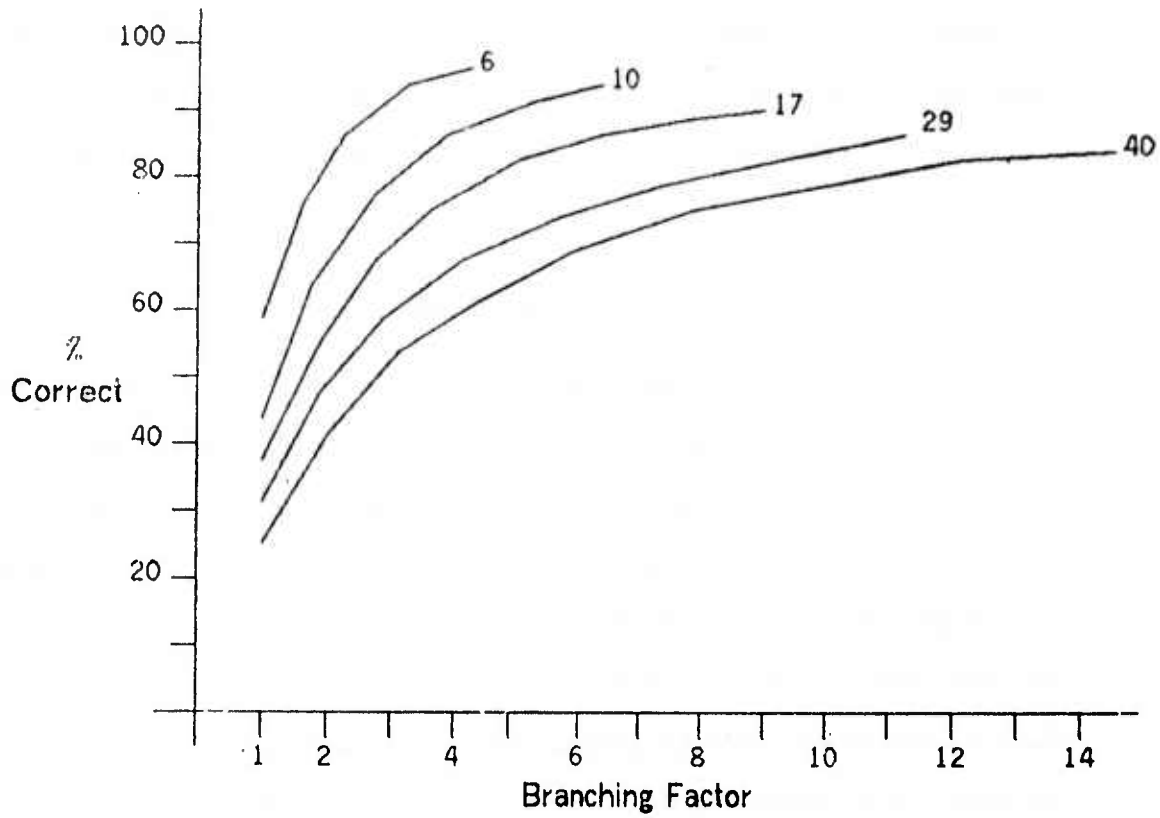


Figure 1: Branching Factor vs. Accuracy of Labeling for Various Target Sets

information content of these representations. The fine spectral parameters seem to contain more of the relevant information about speech. Labeling performance is counter-intuitive, in that it does not seem to matte which representation is chosen. However, this agrees well with the rapidly growing belief that the bulk of the error-inducing ambiguities of human speech are of a higher level nature.

In addition, methods are presented for segmenting and labeling speech in a straightforward, parameter-independent manner. These methods perform well, for their simplicity, when compared with both human and machine results at the same low level. They are not meant as a substitute for higher level knowledge sources, but rather as available tools or lower bounds on acceptable performance for the initial signal-to-symbol analyses. A methodology for evaluating performance, which is closely allied to our view of the separablility of the levels of representation, is offered. A parameter, d', of the signal detection model appears useful as a broad estimate of behavior.

Finally, a viewpoint is put forth concerning the role of pattern classification techniques in recognition processes of this level. It is our belief that more serious application of these methods -- particularly of methods for training which respond to the empirical data, rather than to a priori ideas about speech -- will yield considerable result in the near future.

Application to Hearsay II

The current configuration of Hearsay II has segmentation and labeling of the input parametric representation performed by a separately running program. Hence, there are no interactions with the other knowledge sources. The parametric representation used for most of the Hearsay II research is the LPC spectrum (SPG). Typically, segmentation knowledge acquired by training analysis of one male speaker is used for all other male speakers with no significant degradation of performance. Labeling knowledge is more speaker specific, and training is performed for each speaker, producing a set of about 100 templates for approximately 40 commonly occuring phones. Euclidean (EUC) or euclidean with variance (SIG) distance metrics are used to provide a set of 5 templates (fewer if there are not enough templates close to the input pattern) with which each machine segment is labeled.

The resultant transcription is intended to represent the utterance at an extremely low level. Segmentation is tuned to miss as few segments as possible, and therefore yields a large number of segments which may be considered extra. (These extra segments are very often indications of transition segments between phones or of regions of intra-phone variation.) The multiple template labeling information is also designed to omit as little possibly relevent information as possible. Hence, a major task of one knowledge source in Hearsay II† is to consolidate and select from this input with the aid of speech specific knowledge at the phonetic and phonological levels.

† See Shockey and Adam's paper in this collection concerning PSYN.

References

[EGA64] Egan, James P., Schulman, Arthur I., Greenberg, Gordon Z., Operating Characteristics Determined by Binary Decisions and Ratings," in *Signal Detection and Recognition by Human Observers: Contemporary Readings*, ed. John A. Swets, Pp.172-86, Wiley and Sons, Inc., New York, 1964.

[GOL75] Goldberg, Henry G., "Segmentation and Labeling of Speech: A Comparative Performance Evaluation," Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Piltsburgh, Pennsylvania, December, 1975.

[ITA75] Itakura, Fumitada, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-23, No. 1, Pp. 67-71, Feb. 1975.

[MAR72] Markel, John D., "Digital Inverse Filtering -- A New Tool for Formant Trajectory Estimation," *IEEE Transactions on Audio and Electroacoustics*, Vol. AU-20, Pp.129-37, 1972.

[SHO74] Shockey, Linda and Reddy, Raj, "Quantitative Analyses of Speech Perception: Results from Transcription of Connected Speech from Unfamiliar Languages," Computer Science Department, Carnegie-Mellon University, Presented at Speech Communication Seminar, Stockholm, Sweden, August, 1974.

WORD HYPOTHESIZATION IN THE HEARSAY II SPEECH UNDERSTANDING SYSTEM

A. Richard Smith
Computer Science Department[1]
Carnegie-Mellon University
Pittsburgh, Pa. 15213

January 26, 1976

I. INTRODUCTION

A central problem for speech understanding systems consists of efficiently and accurately determining what words at the lexical level are implied by the data at lower levels. As speech systems permit larger vocabularies and languages with less restricted syntax and semantics, they must depend more on bottom-up methods to limit the search space of possible word sequences. A bottom-up word hypothesizor is presented which uses classes of syllables (called SYLTYPES) to support word hypotheses. A Markov probability model relates a lower level representation (in this case phones[2]) to a sequence of states defining a SYLTYPE. Words are suggested by these SYLTYPES (using an inverted lexicon) and possibly pruned (for multisyllabic words) depending on adjacent SYLTYPES. The definition of SYLTYPES is made, based on the training data and the word vocabulary, to optimize the word hypothesization accuracy. The method, as implemented in POMOW for the Hearsay II speech system, is shown to reduce the search space effectively for a vocabulary of 1000 words.

In this working paper we will cover five topics: 1) background on the problem of word hypothesization, 2) supporting ideas, 3) implementation of these ideas for the HSII system, 4) results to date, and 5) problems to investigate. The work discussed here is in a changing state. Some of the methods have been chosen because they "seem right" and have not been tested thoroughly. Other methods have been selected for their ease of implementation and may yield to better ideas.

---

[1]This research was supported in part by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

[2]In this paper we will use "phone" to refer to a sound detected and classified by a program and "phoneme" for the expectation of that sound as entered in a word pronunciation dictionary.

II. PROBLEM BACKGROUND

A central problem for speech understanding systems consists of efficiently and accurately extracting information from different sources of knowledge and applying it to determine what words at the lexical level are best represented in the data at lower levels (e.g., phones, acoustic segments, parameters).

Two types of strategies for applying this information are top-down strategies and bottom-up strategies. Top-down strategies (also called "analysis by synthesis") determine which is the best word at the lexical level for some region of time by transforming all possible words into the representation of a lower level, comparing each word at that level, scoring each word, and choosing the word with the highest score. These strategies are feasible only in task domains which have small vocabularies and strong syntactic and semantic constraints, and are found in most first generation speech systems. The early Lincoln, SDC, SRI, Sperry Univac and HEARSAY I systems are examples. However, when vocabularies become large and syntactic and semantic constraints become weak, top-down strategies are lost in the combinatorics of potential word sequences. The Dragon speech system (Baker 1975) must be classified as using both a top-down strategy and a bottom-up strategy since it essentially does both at once. The Dragon system does have the same sensitivity to word vocabulary size that top-down strategies have.

Bottom-up strategies, as in POMOW, attempt to infer from the information at the lower level what the words are at the lexical level. The ideal bottom-up strategy would propose only one word at the lexical level (later verified to be correct) for each word-sized region in the lower level representation. Size of vocabulary and degree of syntactic and semantic constraints might effect the efficiency, but not the accuracy, of this ideal bottom-up strategy. It would have to collect and use the right information at the lower level to point to the correct word and only the correct word for each word-sized region of the utterance. Of course the errorful and ambiguous nature of speech prevents such an ideal bottom-up strategy, but we can try to approach this ideal. That is, we can limit the number of proposed words per word-sized region without usually excluding the correct word, by choosing and using the most effective information from the lower level.

The information we use to support word hypothesis and how we group this information into units define what we will call "units of support". Three possible criteria for good "units of support" might be:

1. Robustness - effected little by surrounding speech.
2. Few units relative to word vocabulary size.
3. Informationally rich - each unit proposes few words.

If a unit of support is robust it will reliably indicate the presence of a word or a set of possible words in various contexts and under different speaker conditions. The second criterion prevents the original combinatoric problem of hypothesizing words from being replaced with the same problem of hypothesizing these support units. The last criterion points us in the direction of the ideal bottom-up hypothesizor. Unfortunately criterion one seems to be weak at all levels of representation in speech and criteria two and three tend to vary inversely. By choosing "units of support" we are introducing a new level of representation between the word level and the lower level representation. We want to optimize these criteria to give the most efficient and accurate bottom-up strategy.

Speech understanding systems which have included a bottom-up strategy have used various "units of support". One (rarely used) version of Hearsay I (Reddy, Erman, and Neely 1973) used gross features (such as the "SH" in "Bishop") to retrieve those words from the lexicon that included the particular feature present in the utterance. The 1974 BBN Speechlis system (Rovner, et al. 1974) used any pair of adjacent phones as a unit of support for its bottom-up strategy. In each phone-sized segment several alternative phonemes might be hypothesized. Each unique pair of hypothesized phonemes in every pair of adjacent segments is used to retrieve the set of words from the lexicon containing that pair of phonemes.

III. SUPPORTING IDEAS

What might be the best units of support for a bottom-up strategy? Gross features have few units and tend to be robust but are not informationally rich. Phoneme pairs may contain more information but cannot be considered robust. Two different ideas influenced the selection of the type of units of support for POMOW. The first is from Fujimura (1974) who proposes that the syllable, phonologically redefined, serve as the effective minimal unit of speech recognition. Though he is primarily concerned with their use in a top-down strategy of template matching, his reasons for using the syllable also apply to bottom-up strategies. He defines an ordering of consonantal elements according to their "vowel affinity" (vowels having maximum affinity). A syllable is then the segment from one minimum in the affinity contour of the utterance to the next minimum. He argues that syllables, and especially stressed syllables, are more robust than phonemes. Coarticulatory effects across a syllable boundary, when such a boundary is definable, are much less and more easily handled with phonological rules then are the effects among

phonemes within a syllable. It seems that a syllable contains enough information so that any syllable would propose only a small fraction of the word vocabulary. A possible weakness of using the syllable as a unit of support is the number of units. The number of syllables, as Fujimura defines them, is on the order of a few thousand. We can limit this number, and unfortunately decrease the information content of each, by putting the syllables into equivalence classes; we call these syltypes.

The definition of syltypes also had influence from another source. Baker's (1975) use of a Markov probability model in his Dragon speech system encouraged us to define the syltypes by a state-sequence and use a similar model for deriving them from the underlying phone sequence.

There are two extremes a state sequence definition of syltypes could take which will produce the same number of units. The first is to form a syltype from a short sequence of many different states. For example, we could use a sequence given by a prenucleus state derived at a low amplitude point, a nucleus state at a local high amplitude point, and a postnucleus state at the next low amplitude point. The second extreme lies in the direction of doing more segmentation to produce a longer sequence of states with relatively fewer different states. POMOW's definition of syltypes tends toward this second extreme with segmentation being on the order of phone lengths and the states being equivalence classes of phonemes. The definition of syltypes can be easily changed within this framework by redefining phoneme equivalence classes for the phonemes and specifying legal state transitions. The decision to use this framework has not been completely analyzed and was originally made because POMOW was to use phone hypotheses as input. This is not necessary. In fact one version of POMOW has used lower level acoustic segments as input together with the same definition of syltypes as was used with phone input.

```
A   A-LIKE:  AE,AA,AH,AO,AX
I   I-LIKE:  IY,IH,EY,EH,IX,AY
U   U-LIKE:  OW,UH,U,UW,ER,AW,OY,EL,EM,EN
L   LIQUID:  Y,W,R,L
N   NASAL:   M,N,NX
P   STOP:    P,T,K,B,D,G,DX,
F   FRIC:    HH,F,TH,S,SH,V,DH,Z,ZH,CH,JH,WH
```

Figure 1: Phoneme Equivalence Classes.

The current definition of syltypes is based on grouping the phonemes into seven classes: A-like, I-like, and U-like vowels, liquids, nasals, stops, and fricatives. Figure 1

gives the class membership for the phonemes. Each class contains two states depending on which side of the syllable nucleus the phoneme appears (e.g., phoneme "T" is mapped to a Stop!left state if it preceeds the syllable nucleus). Vowels are also mapped to left and right states. Typical state transitions, found in a 275 word vocabulary, are described by the network given in Figure 2. For example, let the above phoneme classes be represented by the symbols A,I,U,L,N,P, and F respectively. The word "AIRPLANES", with the pronunciation <EH R> <P L EY N Z>, is mapped into the syltypes IL and PLINF.

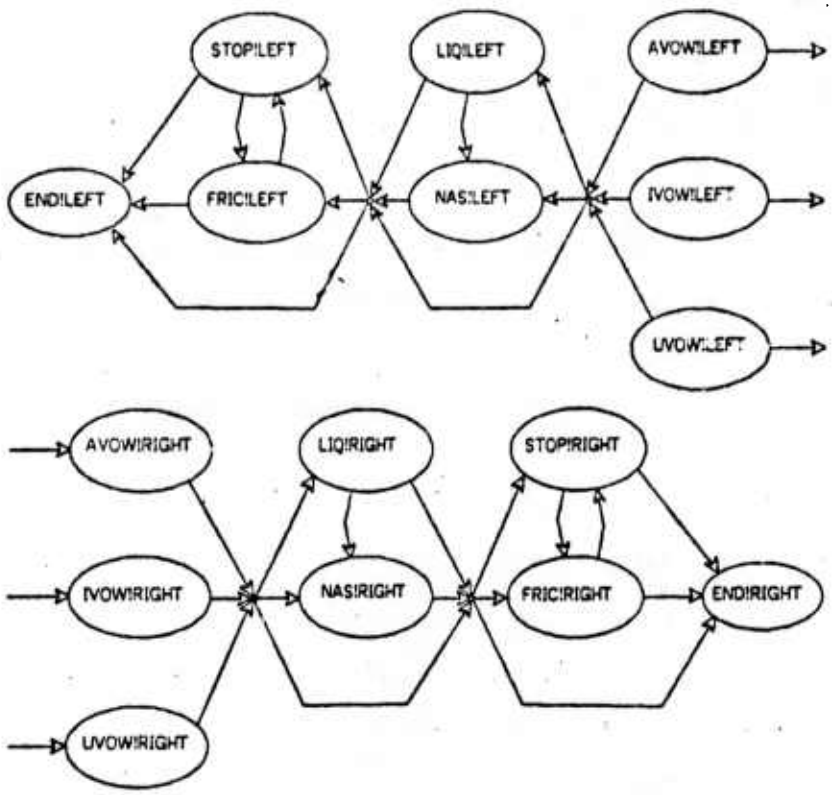


Figure 2: Syltype State Network

How many syltypes does this definition permit? Even with preventing cycles between the Stop and Fricative states (for both the left and right parts of the network), there are more than 900 syltypes. However, the number that are encountered in a fixed vocabulary is usually smaller. The graph in Figure 3 of word vocabulary size versus number of unique syltypes (for five vocabularies we have used) shows that a vocabulary of 1000 words has about 250 syltypes. The syltypes are informationally rich since on the average each will indicate the presence of only a small fraction (2%) of the vocabulary. In the worse case, a particular syltype occurs in 10% of the vocabulary. Figure 4 is a sample from a syltype lexicon.

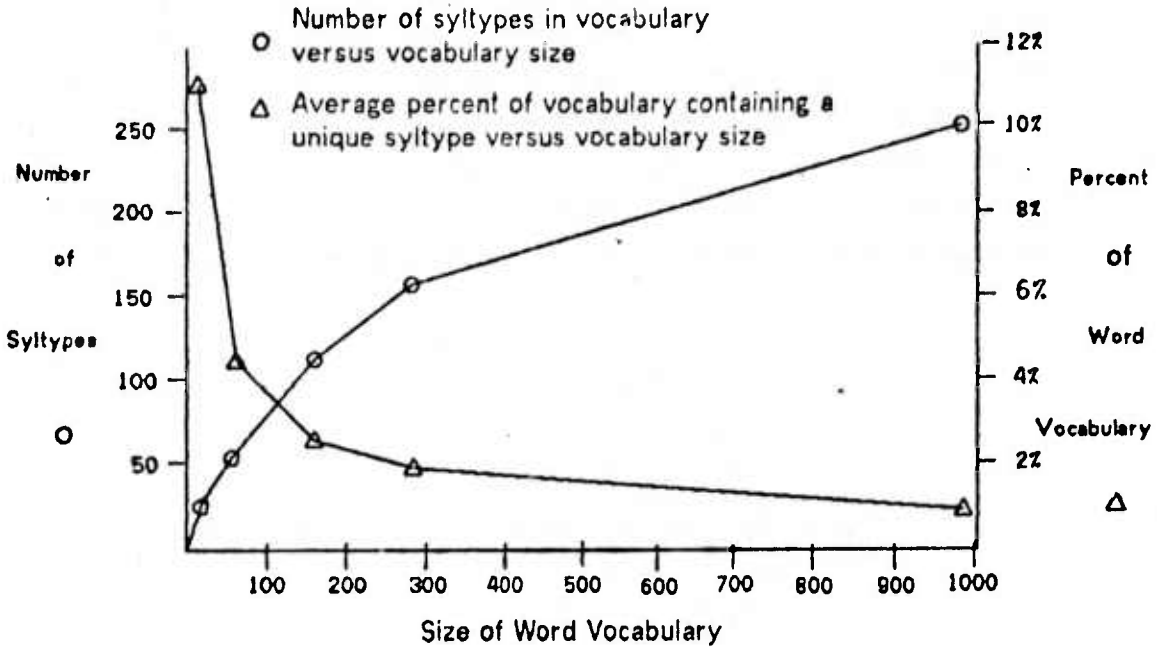


Figure 3: Words versus Syltypes

Syltyp	Stress	Word
AN	1	A
	0	AMERICAN
	1	AND
	0	MUSEUM
	1	ON
	1	ONTARIO
	0	VETERAN
ANP	1	AND
ANF	1	NEW=ORLEANS
AL	1	ALBERTA
	2	ALCOHOL
	1	ALCOHOL
	1	ALL
	1	ARCARO
	1	ARE
	2	ARPA

Figure 4: Sample from a Syltyp Lexicon.

How should we calculate the probability of taking a particular path through the syltype state network given a sequence of phones? That is, how do we assign probabilities to different syltypes? Let Y[1:Q] be a sequence of phones and X[1:Q] be a

sequence of syltype states forming some syltype. We want to find the probability of X[1:Q] given that the phone sequence Y[1:Q] has been detected. This can be calculated by:

Eq(1) $P(X[1:Q]=x[1:Q] \mid Y[1:Q]=y[1:Q]) =$
$\pi_{q=1:Q} P(X(q)=x(q) \mid X[1:q-1]=x[1:q-1], Y[1:Q]=y[1:q]).$

Using the Markov assumption that

$$P(X(q)=x(q) \mid X[1:q-1]=x[1:q-1]) = P(X(q)=x(q) \mid X(q-1)=x(q-1))$$

and the assumption that the information in the sequence y[1:q-1] is sufficiently encoded in the state x(q-1) we have:

Eq(2) $P(X[1:Q]=x[1:Q] \mid Y[1:Q]=y[1:Q]) =$
$\pi_{q=1:Q} P(X(q)=x(q) \mid X(q-1)=x(q-1), Y(q)=y(q)).$

For q=1 we use $P(X(1)=x(1) \mid Y(1)=y(1))$ in the product of the right hand side of Eq(2). The Dragon Speech System uses a similar equation and employs a dynamic programming method to find the word sequence with the highest probability. However, the goals and restrictions of Dragon differ from POMOW. Dragon must determine what one utterance was spoken, so choosing the most probable path in its state network is the best decision. But POMOW has limited knowledge about speech and must work together with other knowledge sources (KSs). It should find a set of most probable paths in its network, i.e., syltypes, each rated so that another KS can decide which one is best. Dragon is restricted from doing this computationally by the number of states needed to describe the whole utterance. With only 16 states (the number of nodes in Figure 2), POMOW can afford to investigate many paths.

To give a more reliable common starting point for finding alternative syltypes, the sequence of phones, and therefore the syltype state network, is not traced left to right (i.e. in the direction of increasing time) but from the syltype nucleus out to the END!LEFT state and then from the nucleus to the END!RIGHT state, as indicated by the arrows in Figure 2. Hence X(1) will be an AVOW!LEFT, IVOW!LEFT, or UVOW!LEFT state.

Once we have syltypes supported by the observed phones, we must find the words which best fit the syltypes. The ideas behind the hypothesization of words from the syltypes are less sophisticated. Each hypothesized syltype suggests a set of words to hypothesize. A particular word is included in the set because it has a syllable which

maps to the syltype and the syllable was marked in the word-phoneme dictionary (currently by intuition) as having enough stress to reliably indicate the word's presence in the utterance. Multisyllabic words in this set are rejected if they match poorly with adjacent syltype hypotheses. The match uses (a yet untested) measure based on the conditional probability that a word's syltype occured given that a particular hypothesized syltype is observed. Words not rejected are rated and put in the HSII data base for other KS's to check.

IV. IMPLEMENTATION

The implementation consists of three programs: a module for the HSII system containing the KS's POM and MOW, (the syltype hypothesizor and the word hypothesizor, respectively), and initialization programs for each, called POMNIT and MOWNIT. Figure 5 illustrates how these programs are connected and used. The implementation is best described by this figure and the figures referenced by it. We will explain some of the more obscure points.

The syntax of the word-phoneme dictionary (Figure 6) permits an AND/OR tree of the possible phonemes in a word. Parentheses and commas indicates an OR group and concatenation of the elements (phones or syllables) indicate an AND group. Angle brackets, "<>", are syllable boundaries, with the number after the opening bracket giving the stress level of the syllable. (We use 0, 1, and 2 to indicate reduced, normal or stressed, respectively, with a default stress of normal). Currently these stress levels are used to indicate to MOW whether or not the word should be hypothesized by the syltype which the syllable will map to. For example, the word "AND" will never be hypothesized by the syltype "IJ" which the syllable "EN" maps to. A "0" in any OR group (whether composed of phonemes or syllables) indicates that the group may be absent. At present many phonological rules have been put into this dictionary as alternate pronunciations.

MOWNIT uses the word-phoneme dictionary and the phoneme equivalence classes to produce a syltype lexicon and a word-syltype data structure. These are basically inversions of each other. The number with each entry in the syltype lexicon (Figure 4) is the stress level; a second number (not shown) associated with each entry is a pointer into the word-syltype data structure (Figure 7). The information in this structure is used by MOW to find the syllable structure of a word and by other KS's for more detailed verification of the word.

POMNIT uses the phoneme equivalence classes to convert phonetically hand-labeled utterances into state sequences. Previous results from the HSII phone hypothesizor are alligned with these states so that frequency counts of [current state,

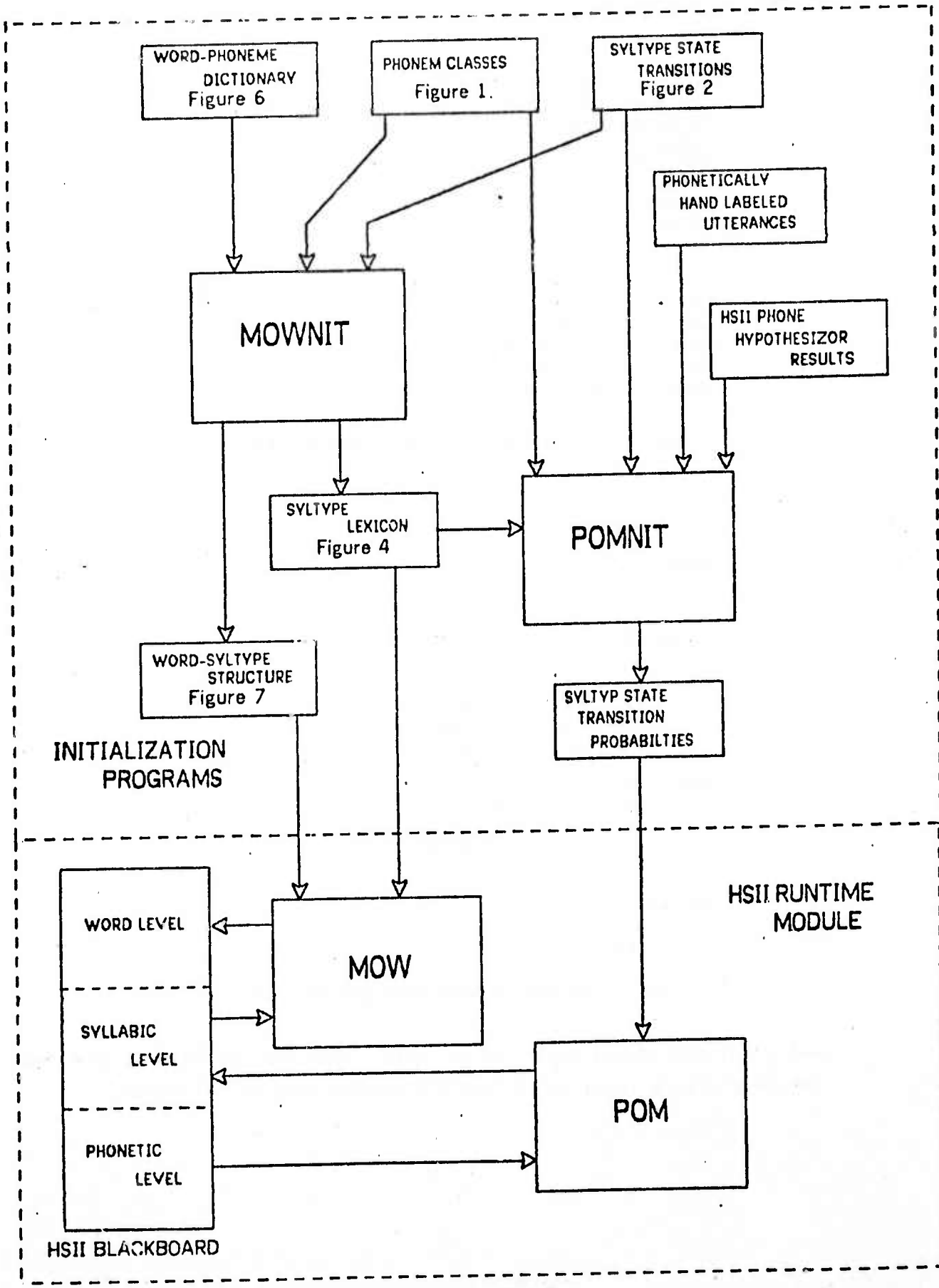


Figure 5: Information Flow

```
ABOUT       (<0AX>,0)<2B AW T >
ACUPUNCTURE <2AE K><0Y(UW,AX)><1P AX NX K><0 (T ,0 )SH ER>
AGRICULTURE <2AE G><0 R (IH,IX) ><1K (AX,AA) L ><T SH ER>
AIRPLANE    <2(EH,EY) R ><P L EY N >
AIRPLANES   <2(EH,EY) R ><P L EY N Z >
AKRON       <2AE ><K R (AX N ,EN)>
ALBERTA     <AE L><2B ER ><(DX,T ) AX>
ALCOHOL     <2AE L ><0K AX><(HH,0 ) AA L >
ALL         <AO L >
ALLIGATOR   <2AE><0L (IH,IX)><1G EY ><0 (DX,T )ER>
AMERICAN    <0(AX M ,EM)><2(M ,0)EH><0R (IH,IX)><K (IX N ,EN)>
ANALYSIS    <0AX ><2N AE ><0L AX ><S (IH,IX' S >
AND         (<0 EN>,<(AE, IX) N (0,D )>)
ANIMALS     <2AE ><N (IH,AX ,EM)><M (AX L ,EL)Z >
ANY         <1(IH,EH)><0N (IY,IH)>
ARCARD      <AA R ><2K EH ><R (AX,OW)>
```

Figure 6: . Sample from a Word-Phoneme Dictionary.

Word	Syltyp	Stress	Pronounciation
ABOUT			
	A	0	AX
	PUP	2	B AW T
ACUPUNCTURE			
	AP	2	AE K
	LU	0	Y UW
	LA	0	Y AX
	PANP	1	P AX NX K
	PFU	0	T SH ER
AGRICULTURE			
	AP	2	AE G
	LI	0	R (IH ,IX)
	PAL	1	K (AX ,AA) L
	FU	1	T SH ER
AIRPLANE			
	IL	2	(EH ,EY) R
	PLIN	1	P L EY N

Figure 7: Example of Word-Syltyp Data Structure.

next state, next phone] triples can be made. These are normalized to give the probability of going from a current state, to a new state given the next phone.

V. PRELIMINARY RESULTS

The results to date are given in Figure 8. For Test I, training and testing was done using 20 utterances which had been phonetically hand-labeled. The training data always excluded the one utterance that was currently being tested, i.e., each utterance was tested based on training on the other 19 utterances. The same method of training was employed for Test II and Test III which used phonetically machine labled utterances (from another KS in HEARSAY II), with vocabularies of 275 words and 970 words, respectively.

	Test I	Test II	Test III
Accuracy:			
% correct words hypothesized:	81%	60%	54%
Average number of words rated better than correct word:	1.0	6.4	9.5
Activity:			
Average number of words generated per utterance word:	25	61	167
Average number of words hypothesized per utterance word:	3.6	13.4	20.9
Training Data Size, # of utterances:	19	28	28
Testing Data Size, # of utterances:	19	12	12
Word Vocabulary Size:	275	275	965

Figure 8: Results.

In the accuracy statistics given above the number of words rated better than the correct word includes one half of the number of words with the same rating as the correct word. Words with the same rating will normally be those supported by the same syltypes. The errors have been analyzed only for Test I. Often (62% of the time) a correct word which was not hypothesized was in fact generated, i.e., supporting syltypes were hypothesized and the word was considered by the module for hypothesization. However the rating of the word (based on its supporting syltypes) was lower than enough other words in the same region to keep it from being hypothesized. (A word is considered to be in the same region as the correct word if its middle third overlaps with the correct word.) If another KS deletes some of these competing words or lowers their ratings, the correct word will be hypothesized. Thus there is a tradeoff between the percent of correct words hypothesized and the number of words rated better than the correct hypothesis.

There are two other causes for POMOW not hypothesizing the correct word. In

15% of the cases the error was caused by insufficient or misleading state transition statistics (to be explained in the next section); 23% of the errors were caused by the phones in the utterance differing too greatly from the phoneme pronunciations found in the word-phoneme dictionary (discussed in the next section).

VI. PROBLEMS TO INVESTIGATE

The results show that when using good phonetic input[1] POMOW currently is able to reduce HSII's search space of possible words for 81% of the words in the utterance from 275 words to 3.6 words on the average and to rate the correct word so that it is in the top two ranks (on the average). For the machine labled phones this statement becomes: POMOW is able to reduce HSII's search space of possible words for 60% (54%) of the utterance from 275 (965) words to 13.4 (20.9) words on the average and to rate the correct word so that it is in the top seven (ten) ranks (on the average). Of course the tradeoff mentioned above permits varying these numbers.

The errors found in Test 1 demonstrate the problems that are encountered when using machine hypothesized phones. The first problem is one of not finding the right syltype when the correct sequence of phones is given. Some of these errors are caused by insufficient state transition statistics and can obviously be reduced by training on more utterances. As with all statistical methods, we must be careful to represent the task domain in the training subset. An example of such an error is missing the syltype "LUI?" because there had been no example in the training utterances that the phone "W" preceding a UVOW!LEFT state should indicate a transition to a LIQ!LEFT state.

The same problem comes from misleading state transition statistics. Whenever a phone must indicate a transition to more than one state from the same state, the correct path through the syltype state network must share the total probability of a syltype occurring with other paths. This problem is common to statistical methods. We are guaranteed to be right the majority of the time but not all of the time. The method will be right all of the time only if there is a mapping of (phone, last-syltype) state pairs onto syltype states. We are just beginning to use a method which minimizes this type of error by automatically defining the best phoneme classes.

The second problem is not finding the right syltype when the sequence of phones for a word differs too greatly from the pronunciations found in the word-phoneme dictionary. We can expect this problem because of the nature of speech. The question is how to best handle it. To some extent we can add alternate pronunciations to the word-

[1] Note that "good" here means a good phonetic transcription of what was actually spoken and not an idealized (dictionary) spelling; thus it does reflect "real speech".

phoneme dictionary but we don't want to enter all possible phoneme sequences for a word that correspond to all possible phone sequences that the phone hypothesizor might generate for the word.

A solution to this second problem has taken the form of defining a metric to match syltypes of words to hypothesized syltypes. For example if the syltype "PAN" is hypothesized there is a significant probability that the syltype "PLAN" occured. The metric has not been tested yet - in fact it may be doing more harm than good at the moment.

There are many remaining areas to investigate. Will stress information permit POMOW to avoid working in regions were it will probably fail? Can we do better by deriving the syltypes from a level of representation lower than the phones? How much training is needed to obtain reliable statistics for our task domain? How much will inter-speaker variability hurt? These and other questions will be investigated as we continue our research.

References

Baker, J. K., "Stochastic Modeling as a Means of Automatic Speech Recognition", Computer Science Department, Carnegie-Mellon University, April 1975.

Fujimura, O., "Syllable as a Unit of Speech Recognition", IEEE Symposium on Speech Recognition, Piltsburgh Pa., April 1974, pp. 148-153.

Lesser, V. R., R. D. Fennell, L. D. Erman, & D. R. Reddy, "Organization of the HEARSAY II Speech Understanding System", IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-23, no. 1, Feb., 1975, pp. 11-23.

Reddy, D. R., L. D. Erman, and R. B. Neely, "The HEARSAY Speech Understanding System: An Example of the Recognition Process," Proc. 3rd Inter. Joint Conf. on Artificial Intel., Stanford, Ca., 1973, pp. 185-193.

Rovner, P., J. Makhoul, J. Wolf, J. Colarusso, "Where the Words are, Lexical Retrieval in a Speech Understanding System," IEEE Symposium on Speech Recognition, Pittsburgh Pa., April 1974, pp. 160-164.

WORD VERIFICATION IN THE HEARSAYII SPEECH UNDERSTANDING SYSTEM

Robert Cronk and Lee D. Erman
Computer Science Department[1]
Carnegie-Mellon University
Pittsburgh, Pa. 15213

January, 1976

Introduction

Because of the difficulties involved in implementing a speech-understanding system, recognition of the words of an utterance is a process which must incorporate knowledge from many different sources, including phonology, prosodics, syntax, semantics, and pragmatics [Newell, et al., 1973]. The Hearsay-II speech understanding system (HSII) uses a multi-level organization with many, diverse, cooperating sources of knowledge (KSs) in an asynchronous, parallel-processing environment. An introduction to this organization is provided in an abstract from Erman and Lesser (1975):

> The hypothesize-and-test paradigm is used as the basis for cooperation among many diverse and independent knowledge sources (KSs). The KSs are assumed individually to be errorful and incomplete. A uniform and integrated multi-level structure, the blackboard, holds the current state of the system. Knowledge sources cooperate by creating, accessing, and modifying elements in the blackboard. The activation of a KS is data-driven, based on the occurrence of patterns in the blackboard which match templates specified by the knowledge source.
>
> Each level in the blackboard specifies a different representation of the problem space; the sequence of levels forms a loose hierarchy in which the elements at each level can approximately be described as abstractions of elements at the next lower level. This decomposition can be thought of as an a priori framework of a plan for solving the problem; each level is a generic stage in the plan. The elements at each level in the blackboard are hypotheses about some aspect of that level. The internal structure of an hypothesis consists of a fixed set of attributes; this set is the same for hypotheses at all levels of representation in the blackboard. These attributes are selected to serve as mechanisms for implementing the data-directed hypothesize-and-test

1 This research was supported by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research.

paradigm and for efficient goal-directed scheduling of KSs. Knowledge sources may create networks of structural relationships among hypotheses. These relationships, which are explicit in the blackboard, serve to represent inferences and deductions made by the KSs about the hypotheses; they also allow competing and overlapping partial solutions to be handled in an integrated manner.

The general "word verification problem" is the following: Given (1) a hypothesis that a particular word was spoken (including some indication of where in the utterance it occurred) and (2) a phonetic transcription of the utterance (which is presumed errorful), determine (a) how likely it is, based on the phonetic evidence, that the word was indeed spoken and (b) where the word's boundaries lie (in the time domain). Other features can be included in the problem specification, e.g., hypotheses about words which are adjacent to the hypothesized word.

In HSII, this problem is attacked with two knowledge-source modules. The first (WOMOS) respells the hypothesized word into expected "surface-phonemic" ("surnemic") spellings, using a dictionary containing spellings for all words in the vocabulary. The second (POSSE) matches these surnemic elements with phones hypothesized from the acoustic input. This paper describes the general issues and strategies of word verification in the HSII system.

Word Hypotheses in HSII

In HSII, word hypotheses are represented as blackboard elements at the *lexical level*. These words may be hypothesized from the phonetic description of the utterance (bottom-up) [Smith, 1976] or from syntactic and semantic predictions based upon a partially recognized utterance (top-down) [Hayes-Roth and Mostow, 1976]. Associated with each hypothesis are *times* (begin, end, and duration) which are used to specify the time region of the utterance in which the word is predicted to appear. Each time specification also has a *range* which is a measure of the uncertainty associated with the time[1]. Special ranges may also be used to indicate: (a) no time information is available for the hypothesis, (b) the time specified is a lower bound for the predicted time, and (c) the time specified is an upper bound for the predicted time.

The connections established between elements at various levels in the blackboard provide implicit structural information. If two or more elements are connected in sequence to a higher-level element, they are considered to be time-

1 E.g., a begin-time of 38 with a range of 3 indicates that the word is predicted to begin between 35 and 41 centiseconds after the beginning of the utterance.

contiguous (structurally adjacent) hypotheses. Elements supporting an option node in the blackboard are considered to cover approximately the same time region of the utterance (competing hypotheses).

In order to accomplish word verification, a dictionary of expected word pronunciations is used. In addition, rules are needed for handling phonological transformations, coarticulation and other contextual effects on pronunciations, word boundary ambiguities, and multiple matching paths when multiple competing plausible phonetic hypotheses exist in the phonetic description of the utterance. The time and implicit structural information (blackboard connections) are also used in verification.

Design of the Word Verifier

In the HSII system, the overall goal may be seen as one of building a consistent representation of the spoken utterance at each of several levels of representation. The inputs to the word verifier consist of such representations at the word and phonetic levels. In order to bridge these levels, the word verifier creates a representation of the word at an intermediate level (called *surface-phonemic*) and attempts to map that representation to the phonetic transcription.[1] The surnemic level contains dictionary phonetic representations of hypothesized words. Modifiers, such as syllable and word boundary markers, may also be present at this level.

The use of the surnemic level is largely extra-theoretical and cannot be easily characterized in terms of general phonological theory [Shockey and Erman, 1974]. However, its use in HSII, as described below, provides a data representation which allows efficient resolution of the ambiguities and problems of word verification.

The process of matching lexical hypotheses to phonetic hypotheses thus becomes one of matching surface-phonemic hypotheses to phones. The approach taken is an incremental one: a small context of several surnemic and phonetic hypotheses is examined and a determination of plausible mappings within that context is made. These results are recorded in the blackboard by creating appropriate links, with ratings, between hypotheses (as described below). The time areas in which these matches are made are thus small compared to the time areas covered by entire word

1 In actuality, an additional level exists in the system between the word and surnemic level: the *syllable level*. This is used primarily by the bottom-up word hypothesizer [Smith, 1976]. The word verifier also uses this level; when respelling words into the surnemic level, they are first spelled at the syllable level and then at the surnemic. For purposes of simplicity of presentation, this paper will not consider the syllable level.

hypothesis. One benefit of this finer granularity of activity is to allow the HSII focus-of-attention mechanism [Hayes-Roth & Lesser, 1976] to schedule selectively the operation of the KSs in different areas of the utterance. Also, the incremental nature of these surneme-phone matches requires less reevaluation of matches when changes in the phonetic data structure or changes in hypothesized word adjacency occur.

Time Information at the Surnemic Level

The problem of limiting the search space for matching dictionary representations of words with phonetic hypotheses is resolved by the use of time and structural information associated with each surneme hypothesis (and which is derived from their associated word hypotheses). For words hypothesized from the phonetic level, a good estimate of the time of the vowel phone for each syllabic nucleus is provided by the bottom-up word hypothesizer [Smith, 1976]. This information -- the begin and end-times -- is carried by the surnemic vowel hypothesis when the word is respelled into its surnemic representation. Only that time area of the phonetic structure corresponding to those begin and end-times need be searched to find a possible match for the surnemic vowel. Finding matches for the other (structurally adjacent) surnemes of the hypothesized words may then proceed outward incrementally from the vowel.

Structural Adjacency (Time-Contiguous) Information at the Surnemic Level

Words hypothesized from higher-level sources of knowledge (syntax and semantics) have limited time information available. At best, the begin or end-time of the word is known, based upon the end or begin time, respectively, of the supported word which predicts the new word hypothesis.

Since the "predictor" word is one which has already received support from the phonetic level, the search for matches for surnemes of the predicted word can use the structural (contextual) information explicit in that support. Just as the word verifier expands outward from the vowel surnemes for a word hypothesized from below (using structural adjacency), it also expands outward in the direction of prediction from the "end" (first or last) surneme of the predictor word using the predicted structural adjacency across the word boundary.

Structural adjacency implies time contiguity. Therefore, in extending outward from an existing match between a phone and a surneme, we need only examine a limited context: those structurally adjacent surnemes and those time adjacent phones in the direction of extension.

Word Verification

Implications of Matches Found

As the matching process proceeds, surneme-phone similarity scores are retrieved from a similarity matrix generated (in part) from training data. If the similarity for matching (linking) is higher than the threshold for that type of match, then the match is made, giving support to the surneme. The similarity score becomes the implication (or confidence rating) for the link. The similarity score is context-independent. However, actual implications given to links may be modified by phonological and surnemic-contextual rules, as described below.

Implications for matches may be either positive or negative. The rating for a word is determined by the implications given to all of the matches found for a sequence of surnemes for the word. When the word-verifier is unable to find a plausible match within the context specified, but rather finds contrary indications, negative implication links are made; these serve to lower the ratings of the word hypothesis.

Multiple Matches of Phones to Surnemes

Since the phonetic representation of an utterance involves overlapping and multiple phonetic hypotheses, the phonetic data structure is really a graph rather than a string of phones [Cole, 1975]. Therefore, multiple time paths through the phonetic hypotheses may provide alternative sets of matches for the sequence of surnemes for a given word (or sequences of phones matched to a surneme). It is not known which alternative match will provide the best overall rating for a word until each path has been matched. Mechanisms exist within the program for duplication of portions of the data structure from the surnemic level to the word level. Several options may exist for matches, and the ratings given a word will depend upon the best match found among the options. Some of the mechanisms used for duplication, and an indication of the decisions made, are shown in the example below.

Contextual Modifications to Implications

As stated above, the similarity score given any surneme-phone match may be context-independent. Coarticulation and other contextual effects, however, are reflected in the (acoustic-) phonetic transcription of the utterance produced by the lower-level knowledge-sources. These effects in changing the pronunciation of words may, in a context-independent matching procedure, severely limit word recognition. In the word-verifier, two mechanisms exist for handling contextual effects on pronunciation:

Since alternative dictionary representations exist, for those words

which are hypothesized from the phonetic level, the pronunciation which most closely resembles the phonetic structure (as determined by the bottom-up word hypothesizer) is hypothesized at the surnemic level.

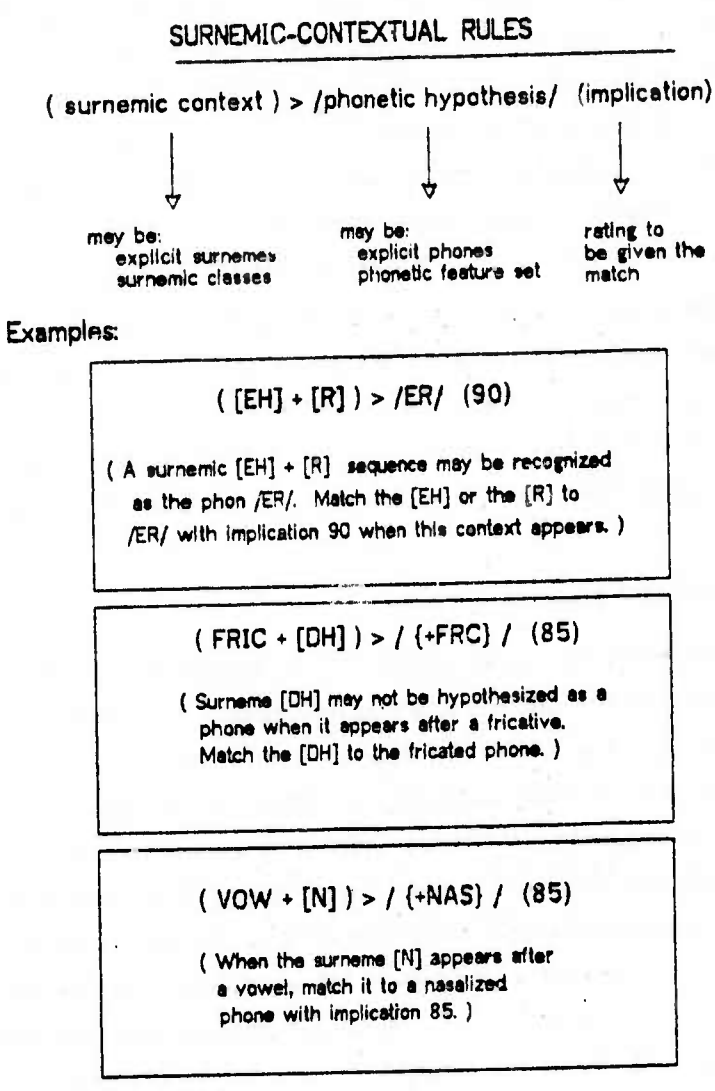


Figure 1. Surnemic contextual rules: syntax and examples.

In addition, phonological and surnemic-contextual rules are applied to determine the final implication for matches. The rules are written in a format which uses the contextual information available at the time matches for a surneme are found. Right- and left-contextual rules are available for implication modification. The syntax for, and examples of, these rules is given in Figure 1. In applying these rules, acoustic and articulatory features of the hypothesized phones and broad class membership of surnemes and phones may be used in determining whether a given rule applies. In addition, explicit recognition rules may be used for commonly observed

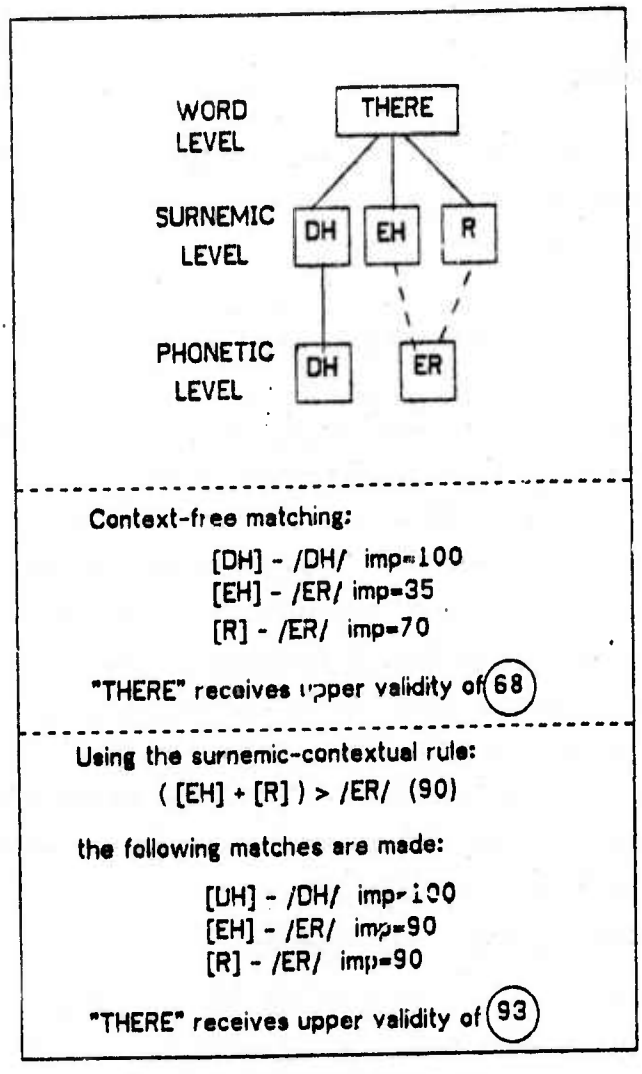


Figure 2. Contextual rule use.

actions of lower-level sources of knowledge. An example of the application of two contextual rules is given in Figure 2. Using only context-independent matching, a rating of 68 would be given the word hypothesis. Application of the contextual rules increases that rating to 93.

Resolution of Word Boundary Ambiguities

In connected speech, word boundaries are not clearly apparent in the phonetic transcription of the utterance. This presents problems in searching for matches for word representations in the phonetic structure. HSII, using structural adjacency to extend matches from established surnemic support, reduces this problem. As words become structurally adjacent (time-contiguous) as the result of higher-level sources of knowledge, word boundary times can be sharpened. Because of the incremental nature of the word-verifier operation, this can be accomplished in most cases without reevaluating matches internal to the word. (The mechanism for doing this matching at word boundaries is identical to that used intra-word; this is a by-product of the fine

grain of the control structure of the word verifier.) The word boundary-times are then propagated upward through the data structure to the word level to aid those upper-level knowledge sources in recognition.

Data Representation and Program Organization

Knowledge source activation is data-directed. In general, a *precondition* for a knowledge source monitors certain *contexts* within the blackboard, and, if patterns specified by that precondition occur, the KS is scheduled for activation. Once activated, the KS may access and change the blackboard, making decisions based upon the knowledge it has about the levels upon which it operates.

Word verification in HSII is performed by four separate knowledge source modules: *WOM* respells word hypotheses into syllables; *MOS* respells syllable hypotheses into dictionary surnemic spellings; *TIME* links phones to surnemes using specific time information associated with the hypotheses; *SEARCH* links phones to surnemes using previously established surneme-phone matches and the structural information explicit in the data structure.

Verifying a Word Hypothesis - an Example

The following example illustrates one way in which a word hypothesis might be verified. In this example (begun in Figure 3), the word "GIVE" has been already "recognized". The word has received support from the phonetic level, and a high rating has been assigned to the word hypothesis. The times shown (<10*3/25*3>) are the begin and end-times and ranges (10 plus or minus 3 csec. and 25 plus or minus 3 csec., respectively) associated with the word hypothesis. The syntax and semantics module has predicted the word "US" to follow "GIVE". Only its begin-time is hypothesized: beginning at time 25 (presumably derived from the end-time of the structurally adjacent "GIVE"). The end-time and both ranges are unknown (shown as "+" in the figure). The following steps show the verification of the word hypothesis "US". Each step represents a separately scheduled activation of one of the KSs described above.

STEP 1: The appearance of a new hypothesis at the word level triggers the precondition of the respelling KS. When the KS is activated, the word "US" is respelled into its dictionary representation, [AX][S], at the surnemic level[1] (Figure 4).

1 Note that, for simplicity, we are ignoring the intermediate process of respelling at the syllable level.

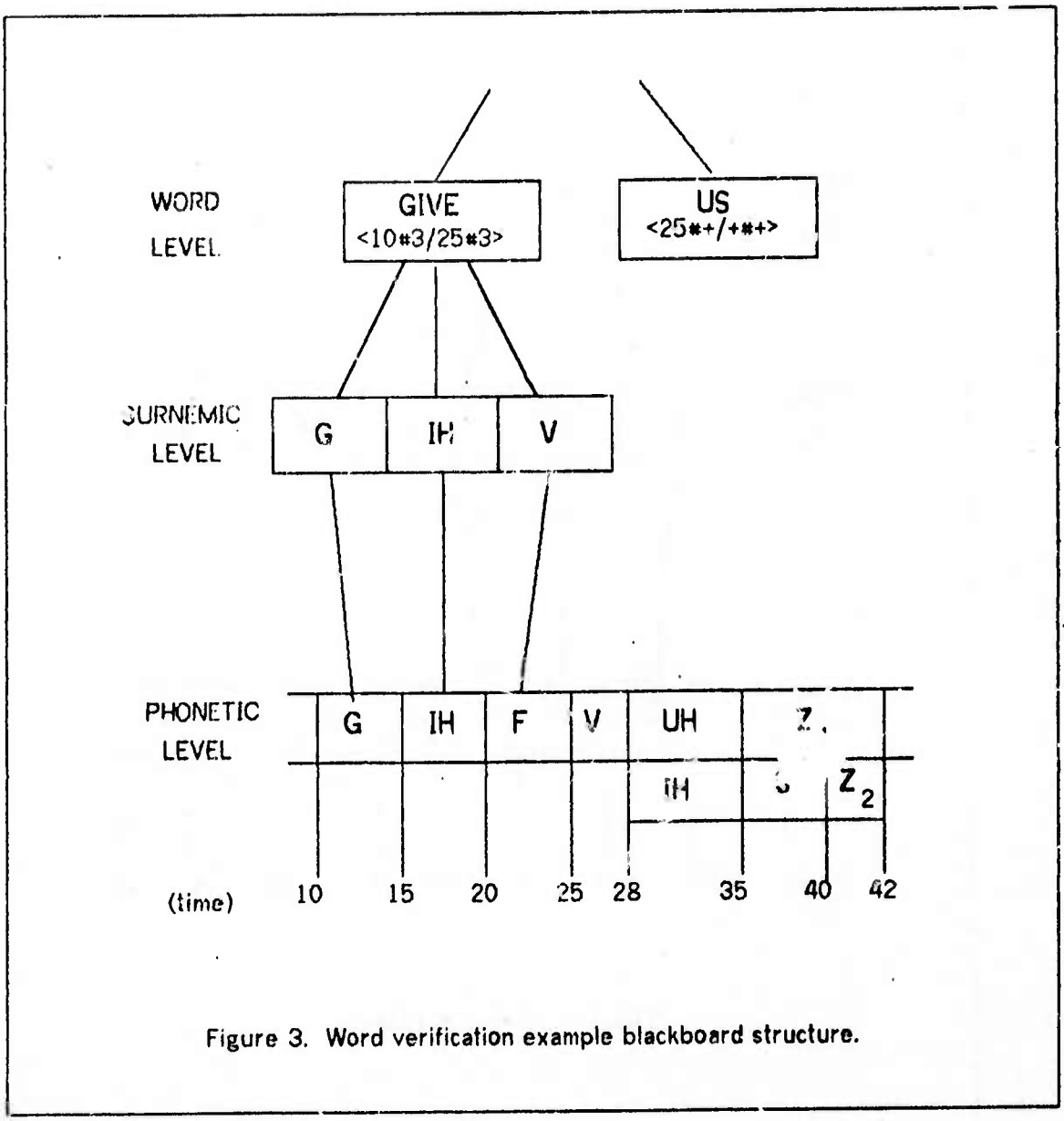


Figure 3. Word verification example blackboard structure.

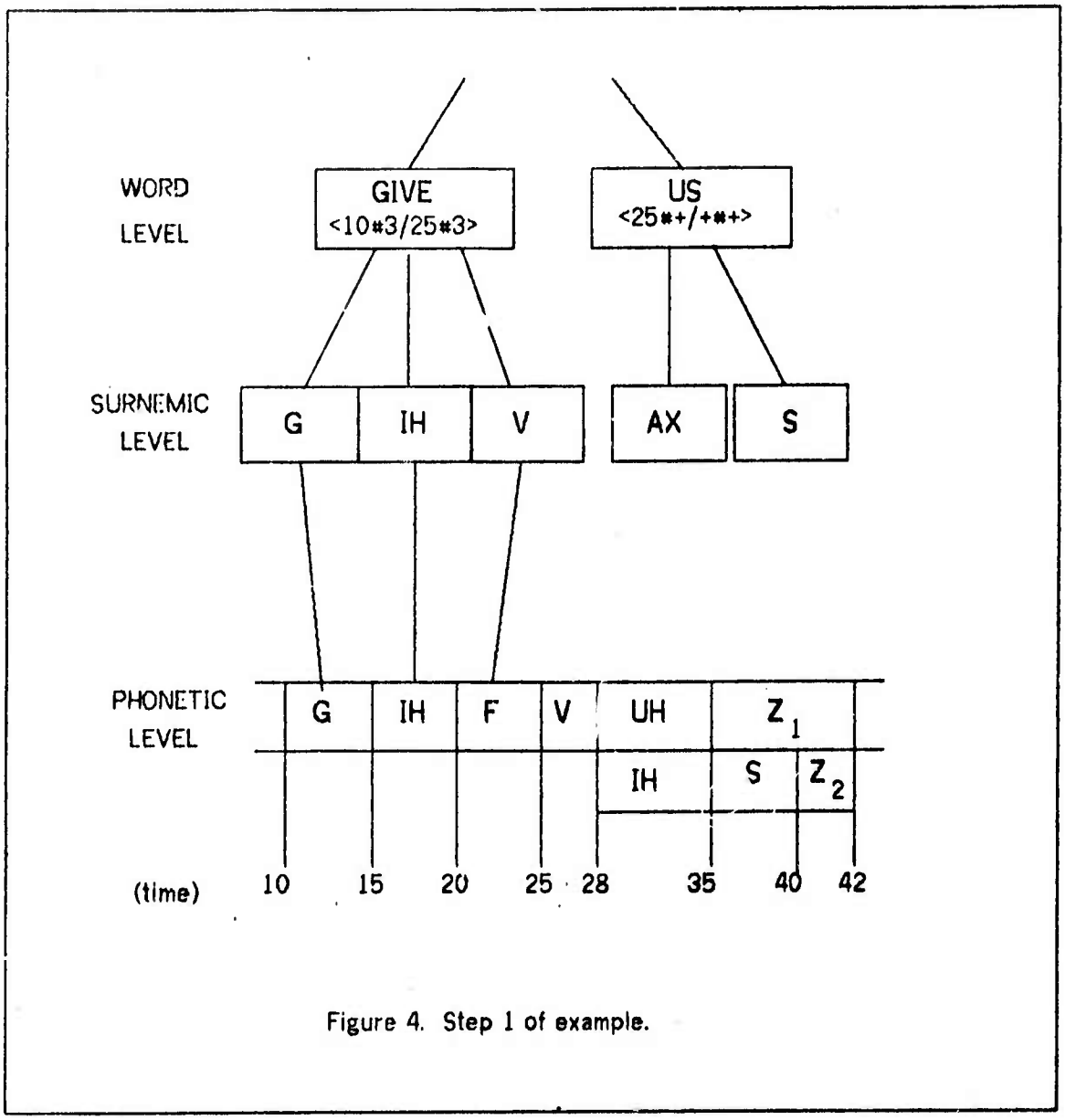


Figure 4. Step 1 of example.

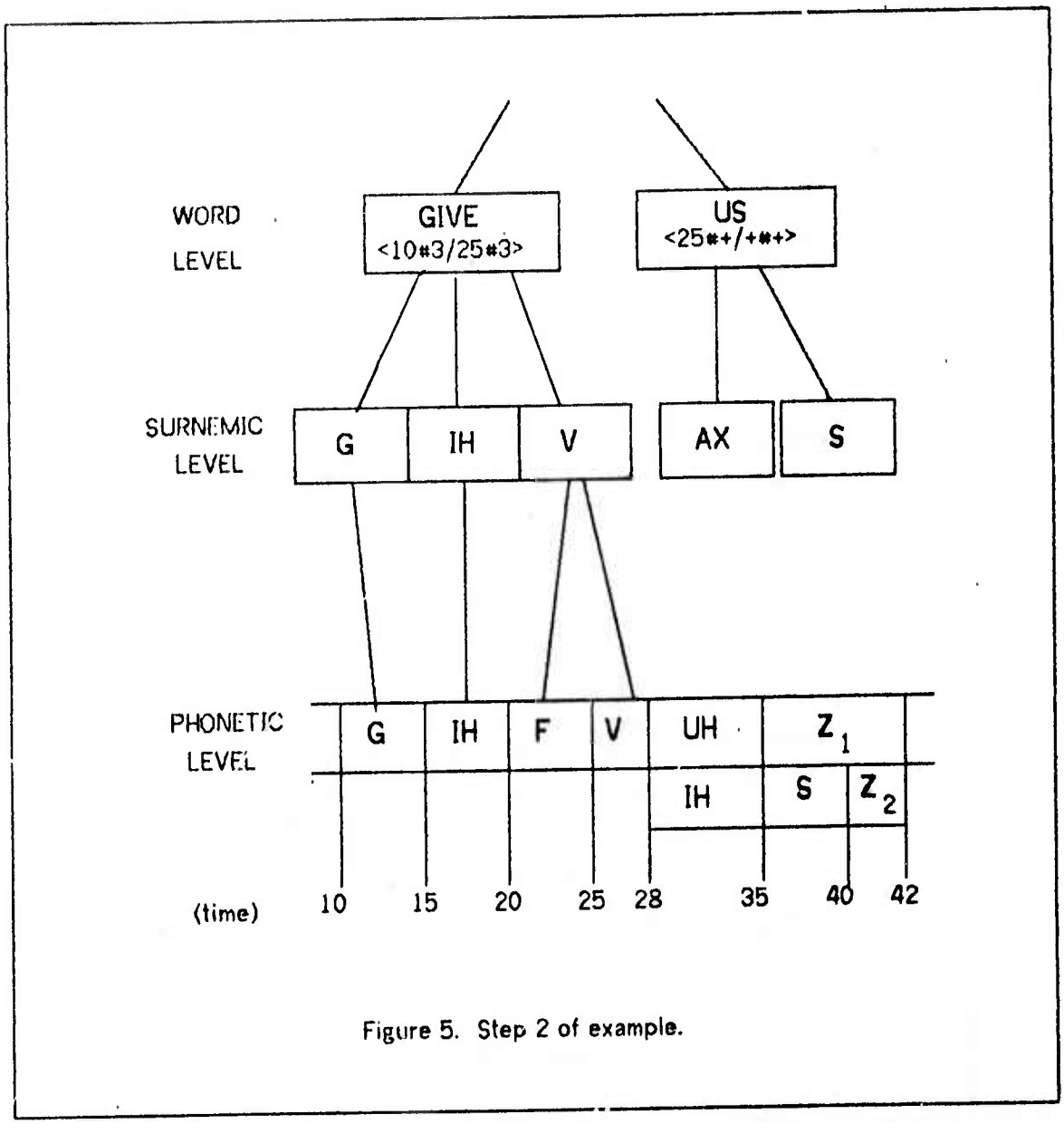


Figure 5. Step 2 of example.

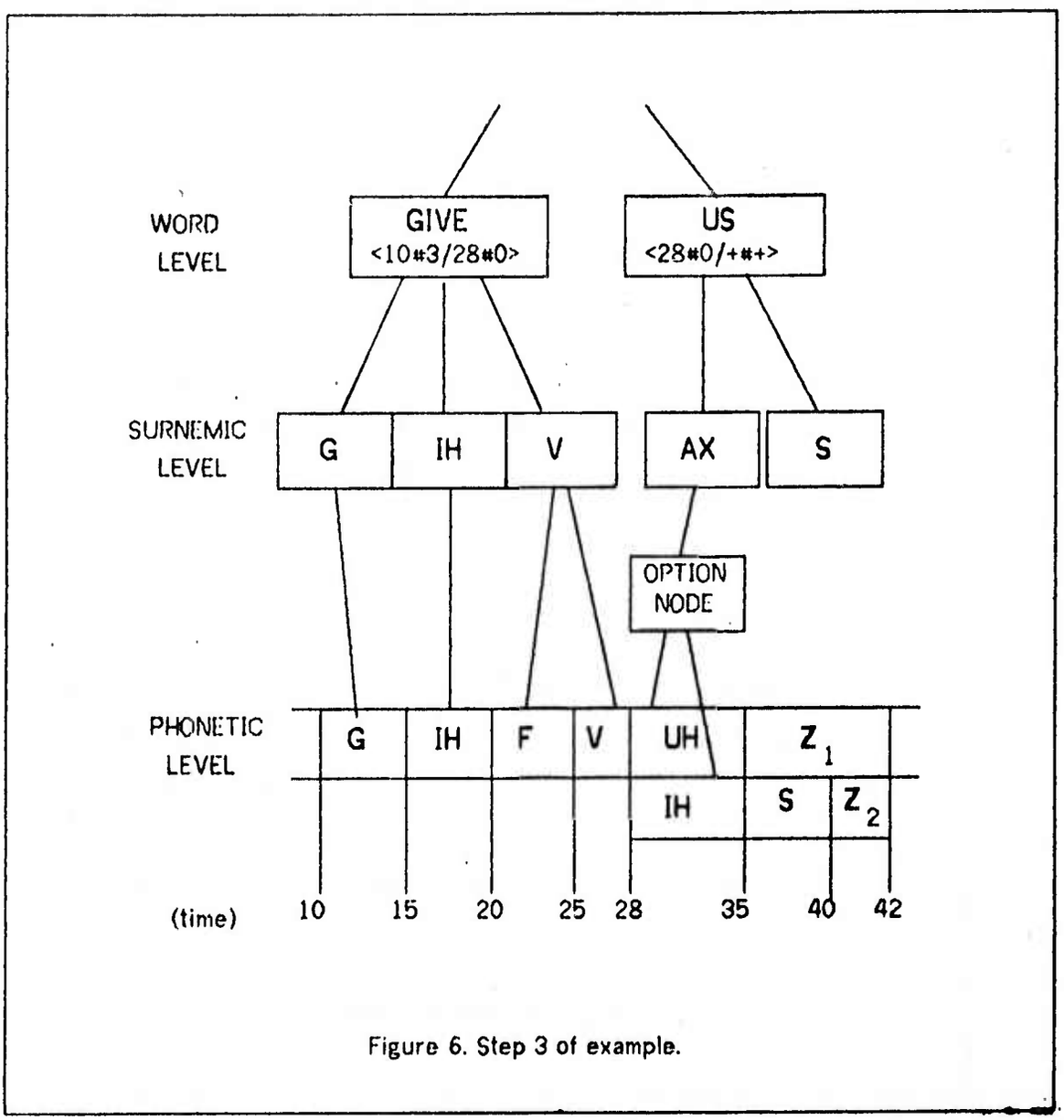


Figure 6. Step 3 of example.

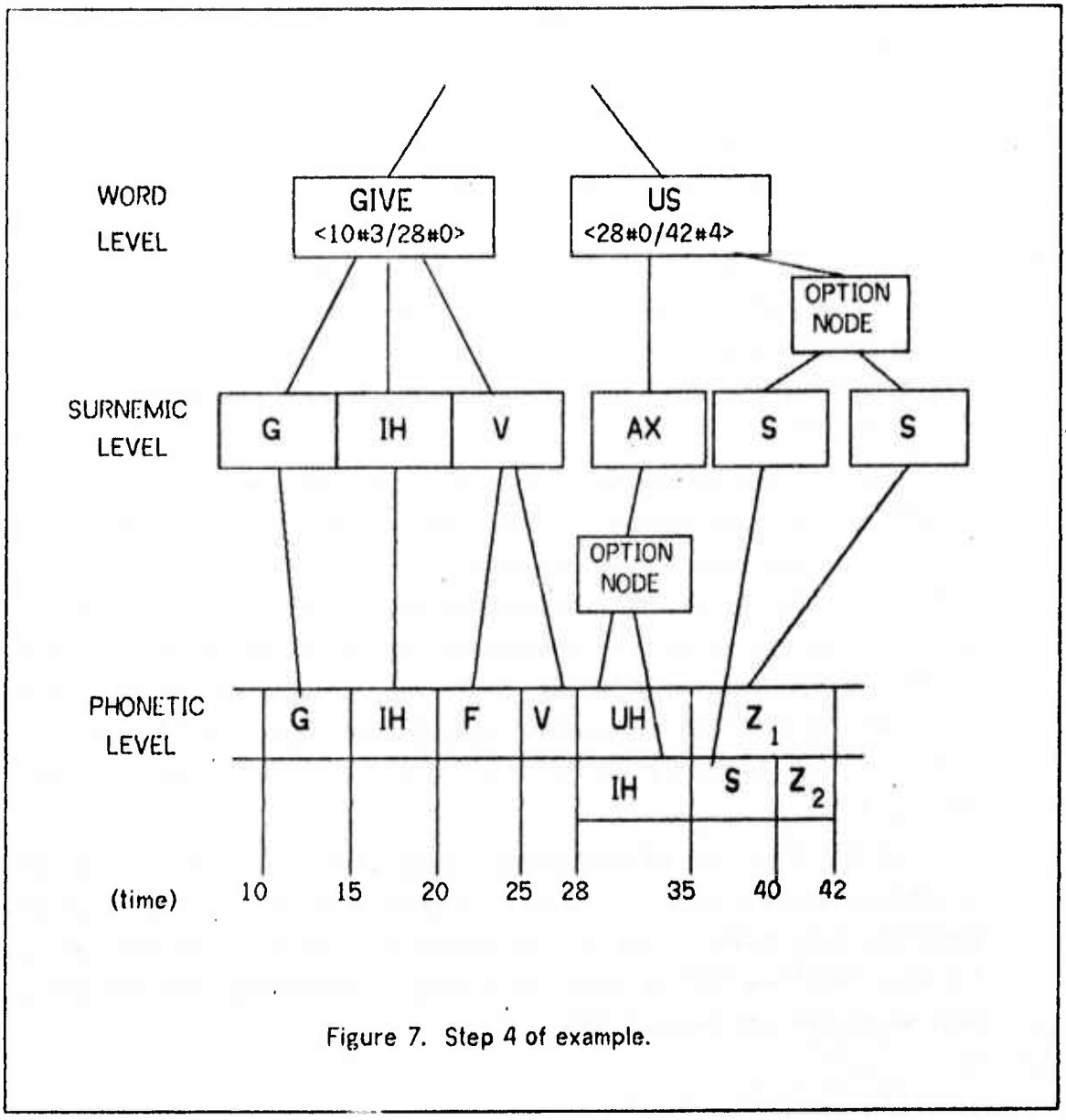


Figure 7. Step 4 of example.

STEP 2: The matching procedure is activated and uses the predicted structural adjacency with the word "GIVE" to extend its search for matches outward from the last surneme of the predictor word ([V]). The context for extending matches includes the "given surneme" (previously matched surneme) [V], the structurally adjacent surneme [AX], the "given phone" (previously matched phone) /F/, and the set of time adjacent phones {/V/}. The implication of each surneme-phone match is considered, and, in this case, the [V]↔/V/ match is made (Figure 5).

STEP 3: In the next instantiation (triggered by the creation of the new link), the context for matching is the given surneme [V], the given phone /V/, the adjacent surneme [AX], and the set of adjacent phones: {/UH/,/IH/}. Two matches are decided upon: [AX]↔/UH/ and [AX]↔/IH/. Since both phones have the same begin and end-times, an *option node* is created at the phonetic level and supported by both phones. The surneme [AX] is linked to the option node. The implication given this link is considered to be the highest implication from the phone options. The word boundary time between these two words is now known, and is propagated upward to the word level (Figure 6).

STEP 4: With the next KS instantiation, two matches are possible: [S]↔/Z_1/ and [S]↔/S/. Since these phones have different end-times, they represent possibly separate matching paths through the phonetic structure. We do not yet know which path will provide the best overall match; therefore, an option node is created at the surnemic level and the surneme is duplicated. One of the options of the surneme is matched to /Z_1/, the other to the /S/. We now have some idea for the end-time of the word "US" and that time is propagated upward to the word level. A range is given that end-time to reflect the degree of uncertainty about what the final end-time will be (Figure 7).

At this point, the matching process cannot continue since there is as yet no structurally adjacent surnemic context to consider. A phonetic representation of the word has been found (though not yet completed[1]). The word boundary ambiguity between "GIVE" and "US" has been resolved, and reasonable begin and end-times have been associated with the word "US".

1 There is still another match to be made: one of the options of [S] to the phone /Z_2/. This will occur when another word is hypothesized structurally adjacent to the right of "US" and the matching process continues from the link [S]↔/S/.

Preliminary Results

System runs for performance evaluation of word verification were made over eleven training utterances spoken by a single speaker using a 275-word vocabulary. Syntax and semantics knowledge sources were not present; all word candidates were predicted from the phonetic hypotheses.

In the eleven utterances, 708 words were hypothesized; 30 of these were the "correct" words (i.e., the actual word spoken hypothesized in the correct time area of the utterance). The results of this very preliminary evaluation of the word verification process are summarized in Figure 8.

Percent of total hypothesized words verified	58%
Percent of correct words verified	97%
Percent of time correct word was highest rated in its time area	48%
Percent of time correct word was in top 5 rated words in its time area	82%
Average number of incorrect words rated higher than correct word in its time area	2.3

Figure 8. Preliminary results of the word verifier.

Improved performance of the word verification process is expected both in terms of decreasing the total percentage of incorrect words verified, and also in terms of increasing the ratings of correct words. Evaluation of performance has only begun. Better training procedures for the phone-surneme similarity matrix will be defined, more efficient matching thresholds for increased discrimination will be determined, and surnemic-contextual rule determination may be automated during system training.

Research is continuing. We feel that we have a viable design for word verification which will allow us to pursue major issues in depth.

References

Cole, A. [1975]. Rule directed phone-phoneme matching. Memo no. ERL-M543, Electronics Research Laboratory, Univ. of California, Berkeley, Sept., 1975.

Erman, L. D. and V. R. Lesser [1975]. A multi-level organization for problem solving using many, diverse, cooperating sources of knowledge. **Proc. 4th Inter. Joint Conf. on Artificial Intelligence**, Tbilisi, Georgia, USSR, Sept., 1975, 483-490. (Also, this volume.)

Hayes-Roth, F. and Mostow, D. J. [1976]. Syntax and semantics in a distributed speech understanding system. To be presented at 1976 IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia. (Also, this volume.)

Hayes-Roth, F. and Lesser, V. R. [1976]. Focus of attention in a distributed-logic speech understanding system. To be presented at 1976 IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia. (Also, this volume.)

Newell, A., J. Barnett, J. Forgie, C. Green, D. Klatt, J. C. R. Licklider, J. Munson, R. Reddy, and W. Woods [1971], **Speech Understanding Systems: Final Report of a Study Group**, pub. by North-Holland (1973).

Shockey L. and Erman, L. D. [1974]. Sub-lexical levels in the Hearsay II speech understanding system. In Erman, L. D. (Ed.), **IEEE Symposium on Speech Recognition - Contributed Papers**, Carnegie-Mellon Univ., Pittsburgh, Apr., 1974, 208-210.

Smith, A. R. [1976]. Word hypothesization in the Hearsay II speech system. To be presented at 1976 IEEE International Conference on Acoustics, Speech, and Signal Processing, Philadelphia. (Also, this volume.)

THE PHONETIC COMPONENT OF THE HEARSAYII SPEECH UNDERSTANDING SYSTEM

Linda Shockey and Christina Adam
Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa. 15213
January 1976

INTRODUCTION

This paper is a description of the linguistically-related principles behind PSYN, the current phonetics module in Hearsay II. It serves as one of many interactive knowledge sources, the main purpose of which is to make easier the mapping of relatively broad phonetic spellings from a dictionary to a (faulty) narrow phonetic transcription made by an automatic segmenter and labeler. PSYN attempts to produce the closest thing to a phonemic transcription that it can without higher-level knowlege. Although each knowlege source in HSII has the potential to interact with any other module, we will make the following explanations as if it were a bottom-up system, for ease of understanding.

Input to PSYN: One or more labels is assigned to each acoustic segment in the speech input as determined by an automatic procedure. Each label is assigned a rating indicating the degree of success of a pattern match which produced it. These, plus amplitude maximum and minimum locations, are the information which is available to PSYN. We hope to include reliable voicing and frication detectors, which will give us additional cues, but at present we work only on labels, durations, and max-min information.

Changes Performed By PSYN: There are two basic kind of transformations which we apply to these input data: 1) mapping onto a feature space and 2) reassigning boundaries and labels. Each input segment may have from one to n labels. But the information given us by these labels is often difficult to interpret. While each label may supply important information, it is sometimes unclear what a particular configuration of them means. For example, suppose we get the labels UW, AA, L, and M for a single segment. We then know that the segment involved has a strong low-frequency component which may make it look like a consonant, but that some high-frequency information is in evidence. We can assume it to be a back vowel if it is a vowel, but as a whole, the phonetic identity of the segment is difficult to evaluate. We try to ameliorate this situation by decomposing each of the labels into features, then trying to make a better guess at what segmental quality is actually present by which features are most strongly represented. We use a set of 13 articulatory and acoustic features, modeled on past feature systems, but somewhat different. We are able to quantize the strength of each feature in two ways: a table indicates how strongly

each feature is present in each element in the ideal case; and the rating on the segment supplies us with a weighting factor. Sounds are represented after this transformation, then, as a weighted feature vector with associated max-min information. Since they are thus decomposed, it is easy the write the phonetic rules described below with reference to classes. The vectors are also available to the phonological component, so phonological rules can be expressed similarly. Note that the features we use are extracted completely from labels at present. Therefore, this scheme bears little resemblence to those which try to detect features directly from some representation of the speech signal.

All of the transformations described in 2) are predicated on sets of features, rather than labels. Feature vectors are mapped back into labels before they are output. Reassigning boundaries and labels is the transformation usually thought of as acoustic-phonetic rules. To do this, we use information derived from sequences of labels and from single labels in some cases. There are three kinds of transformations which take place here: sequential, splitting, and positional relabeling.

Sequential transformations involve combining of adjacent fine phonetic segments to make larger segments. A well-known example is the combination of silence + optional burst + aspiration into a voiceless stop. If we receive a sequence of labels:

time	segment
40-45	- (silence)
45-51	S

we will create the hypothesis that a voiceless stop, probably /T/, exists in the time span between centiseconds 40 and 51. Of course, the original hypothesis put forth by the segmenter and labeler is still present, in case our combination of these segments is in error. Duration of the elements involved is considered in this rule, as in most. A less commonly-discussed but very commonly-found articulatory event is the splitting of a voiced stop into two portions as a result of back pressure behind the closure. The voiced stop begins voiced and therefore is assigned a label of, e.g., M, D, or V. Then the voicing drops off which produces a silence. This is interpreted as a new segment by the automatic devices. Therefore, there is a rule in PSYN which combines sequences such as:

time	segment
40-45	M,B
45-50	-

into a voiced stop and assigns it the label most congruent with its place-of-articulation features. Other combining rules in PSYN attempt to identify short vowels "on the shoulders of" long vowels as transitions and to combine the transitional silences which occasionally occur with fricatives with their accompanying /S/, /F/, etc. We have worked on diphthong rules, but have met with little success.

Splitting transformations make a single segment into two (or theoretically more). The only rule of this type which we have at present takes a long, highly velarized, vocalic segment and splits it into vowel + L, with appropriate durations.

The third type of transformation is positional relabling, which may be considered equivalent in part to allophony rules. For example, if we find a highly velarized vowel which a) has no local maximum and b) is next to another vowel, we hypothesize it could

be W or L; if it meets the above requirements but is retroflex instead of velarized, we call it /R/.

In a very broad sense, then, PSYN accepts a set of rated labels and max-min information, maps the labels into a feature space, makes new hypotheses based on sets of features, and outputs a new set of labels congruent with those hypotheses. While our rate of correct output labels remains less than 40%, we find that, given the best path through the various alternative hypotheses, we reduce the number of extra segments generated by the automatic segmenter by about 50%. When 11 larger classes (high vowels, low vowels, fricatives, nasals, etc.) are considered, the percentage correct averages around 75.

IMPLEMENTATION

The bottom-up processing of HearsayII begins with an acoustic segmentation and labeling. On the basis of this acoustic information, the phone synthesizing module (PSYN) provides a phonetic transcription of the speech signal. Other sources of knowledge then use this transcription to hypothesize syllables and words.

PSYN is composed of two major sources of information. The first source is an acoustic representation of the speech signal. Feature vectors are used to describe the labels which are assigned to the acoustic segments. Amplitudal information is also provided in the form of MXNs which are hypothesized at the segmental level. The second source of information present in PSYN is a set of phonetic rules. Rules written in a similar manner to those of traditional phonology are read by PSYN and are applied to the segmental hypotheses. Phonetic hypotheses are made as indicated by these rules.

Low-Level Combining: There are two HearsayII knowledge sources within the PSYN module; CSEG and PSYN. A preliminary pass is made on the segments by the low-level combiner called CSEG. CSEG is invoked by a set of unused segmental hypotheses which have a left and right context. The knowledge source then acts on each segment individually. The time boundaries, acoustic labels, and a weighted feature vector formed from the labels of the segment are accessed from the data base and lexicons. A feature comparison is made between the segment and each context. If the two segments differ in any feature by more than a given threshold, a low level combination is not performed. In addition, two segments are not combined if both have a local maximum in amplitude. If the features of any two or three adjacent segments are close enough to pass both comparison tests, a combining operation is performed. Labels for the new segment are obtained by determining the best matches between the averaged features of the segments and the labels in the segmental feature file. The best labels are combined to identify the new segment. Ratings are based on the distance from each chosen label to the averaged features. The new feature vector is formed from the averaged features of the individual segments which form the new segment.

Phone Synthesis: After the low-level combining operation is attempted, the phonetic processing of the segment. is performed by the other phonetic knowledge source, PSYN, which also requires a segment with a left and right context. The feature vectors

of the segments involved are used to represent the acoustic information. A series of phonetic rules is used to test the segment and its contexts. If the required conditions are met for any of these rules, phones of the appropriate class are proposed. If none of the phonetic rules apply, a relabeling process based on the set of all phones is performed. The set of phones proposed in each case is chosen by comparing the weighted feature vector of the segment with the feature vector of the labels in the desired phone class. Those phones found to be the closest are selected. The best match is determined by a euclidean-type distance measure between the two vectors. A rating routine assigns an implication to all possible links, based on the euclidean distance measure and a multiplicative factor assigned to each link within the rule.. After all possible phones for a segmental triple have been proposed and rated by the individual actions, the best (highest rated) phones are hypothesized.

PHONETIC RULES

The phonetic rules used by PSYN are written in production system format. The condition part of the rule is defined by a list feature requirements which must all be satisfied for the rule to apply. Feature tests for each segment are expressed in terms of a feature name and a threshold level. Both the feature names and the threshold levels are predeclared by the module.

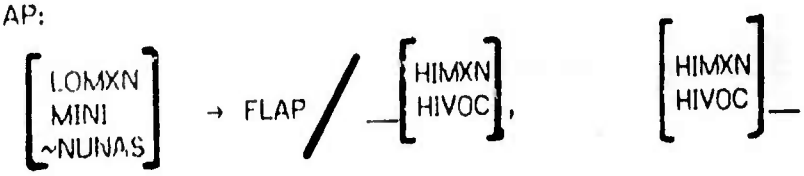
The action portion of the rule is defined by a procedure call. Each rule action defines a subset of phones for relabeling, performs the relabeling based on feature vectors, and rates the resulting labels.

There are three categories of phonetic rules implemented in PSYN. The simplest rules are relabeling rules. A single segment is matched with the phone labels in a subset specified by the rule. The phones hypothesized will span the same time interval as the segment. Context links may be formed if relevant. A more complex rule type is a combining rule. Two or three segments are combined to form a single phone which spans the time interval of all segments involved. Occasionally PSYN may determine that a segment is missing at the lower level. The third type of rule form splits such a segment into two phones.

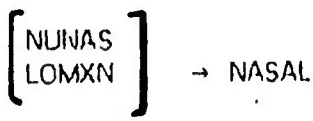
Explanation of Features: High, mid, and low (HI,MID,LO) refer to degree of closure. It is greatest for stops, least for open vowels. Front, central, and back (FRNT,CENT,BK) refer to point of articulation or, for vowels and glides, greatest point of approximation. Rounded, retroflexed, verarized, voiced, fricative, vocalic, consonantal, and diphthongal (RND,RTR,VEL,NAS,VCD,FRC,VOC,CON,DIPH) are used in their usual senses. Hiamp (HIA) indicates a segment which is expected to have relatively high energy, as for example [s] as opposed to [f]. Null (NUL) indicates very low-energy segments. Low frequency (LOF) is a feature shared by all segments with a concentration of energy at the bottom of the spectrum, e.g. nasals, l, and uw. "MXN" describes an actual amplitude contour as discovered by lower-level processes. Prefixes such as "NU", "HI", etc. indicate threshold levels, which are set by the user.

RELABELING RULES

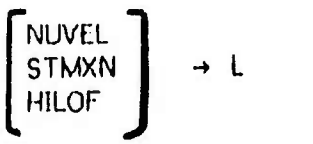
FLAP:

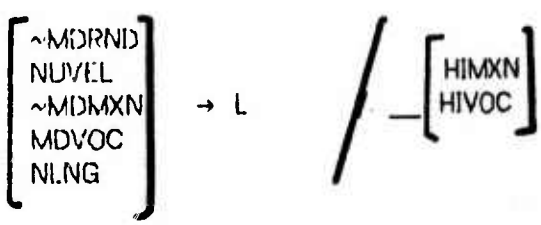


NASAL:

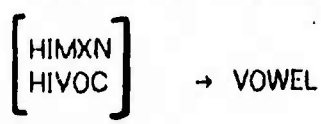


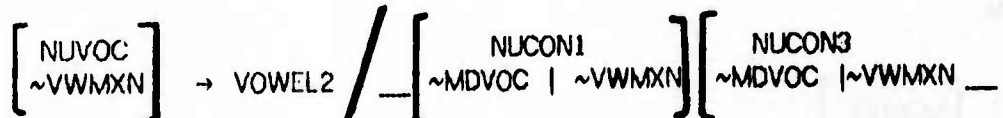
L:

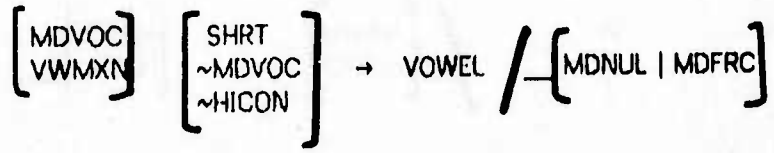




VOWEL:







CONSONANT:

[MDCON, ~MDMXN] [HICON] → CONSONANT

SYLLABIC RESONANTS:

[HIVEL, NUCON, HIMXN] → EL

[HIRTR, SLCON, HIMXN] → ER

[HINAS, NUCON, HIMXN] → EN | EM

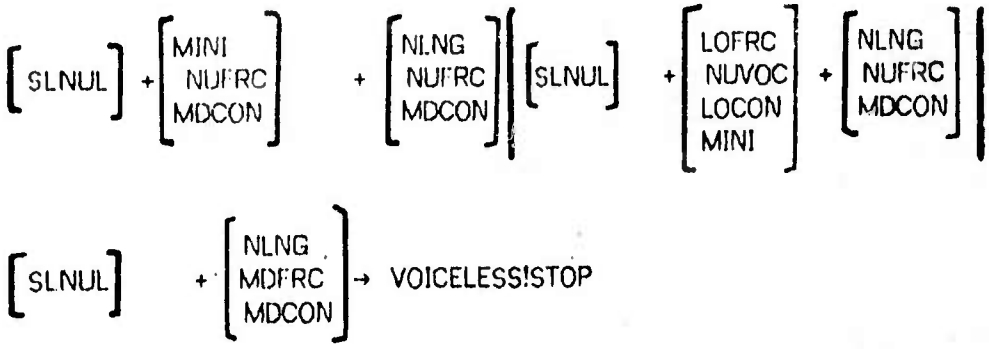
[MDNAS | MDVEL | MDRTR] → SYLRES / __ [HIFRC] [HIFRC] __

W:

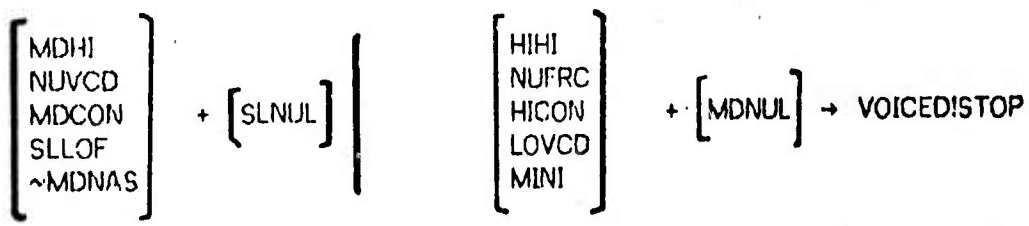
[MDRND, NUVEL, ~MDMXN, MDVOC, NLNG] → W / __ [HIMXN, HIVOC] [HIMXN, HIVOC] __

COMBINING RULES

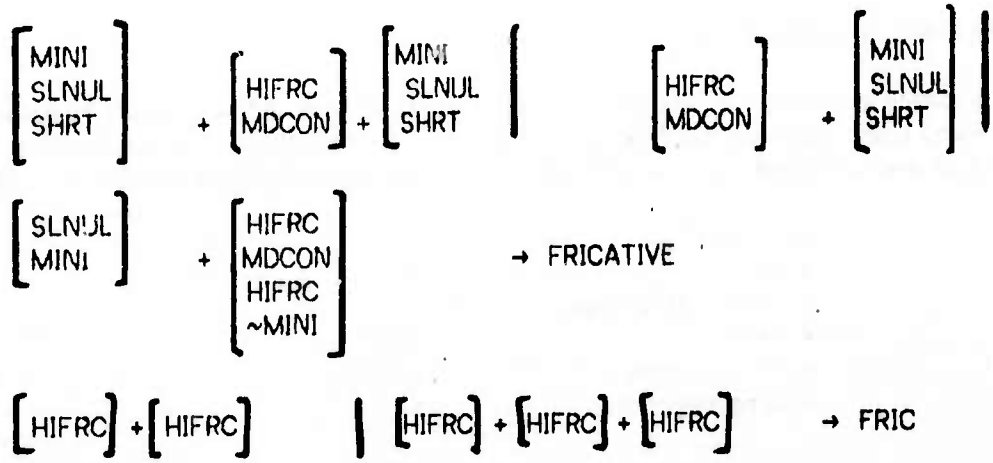
VOICELESS STOP:



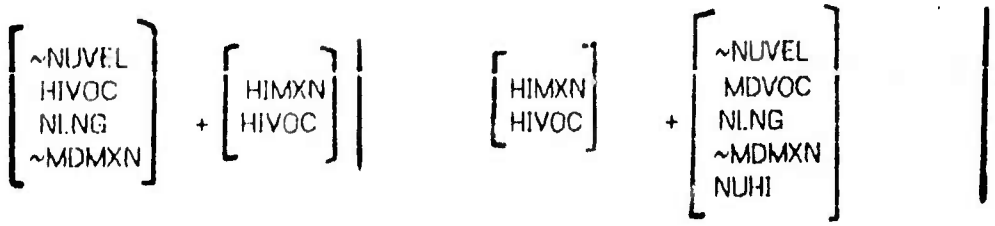
VOICED STOP:

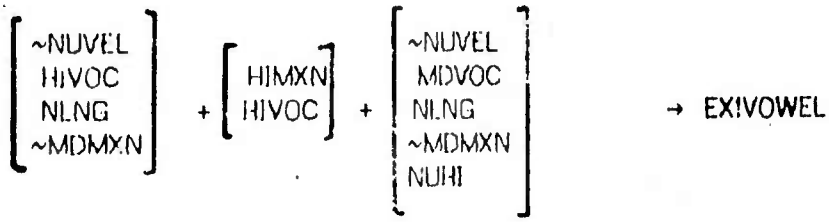


FRICATIVES:

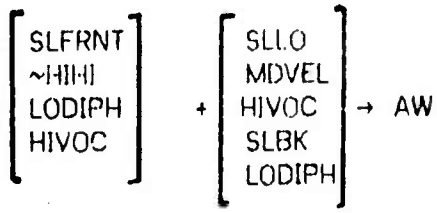


VOWEL:



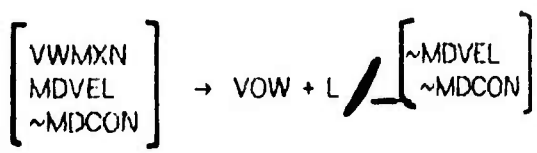


AW:



SPLITTING RULES

L:



PERFORMANCE EVALUATION

Performance evaluation is done by a program which compares the phonetic hypotheses to a hand transcription. The best path of phones is matched with the hand phones and statistics are collected. The current version of PSYN produces the following results:

total matched	611
% correct	29
% correct 1st choice	23
# labels	2089
extra segments	268
missing segments	41

APPENDIX: Confusion Matrices

Confusion matrices based on all labels and smaller label sets are also available. Three phone label subsets are generally used for evaluation purposes:

CL1:

```
HIGH=IY,IH,IX,UW,UH
MID=EY,EH,OW,AX,ER
LOW=AE,AA,AO
FRNT=IY,IH,EY,EH,AE
CNT=IX,AX,ER
BAK=UW,UH,OW,AO,AA
DIPH=AY,AW,OY
LIQ=L,R,EL
GL=W,Y
NAS=M,N,NX,EM,EN
VOST=B,D,G
VLST=P,T,K,Q
FLAP=DX
VOFR=DH,V,WH,Z,ZH
VLFR=TH,F,S,SH,HH
GRB=!,&
```

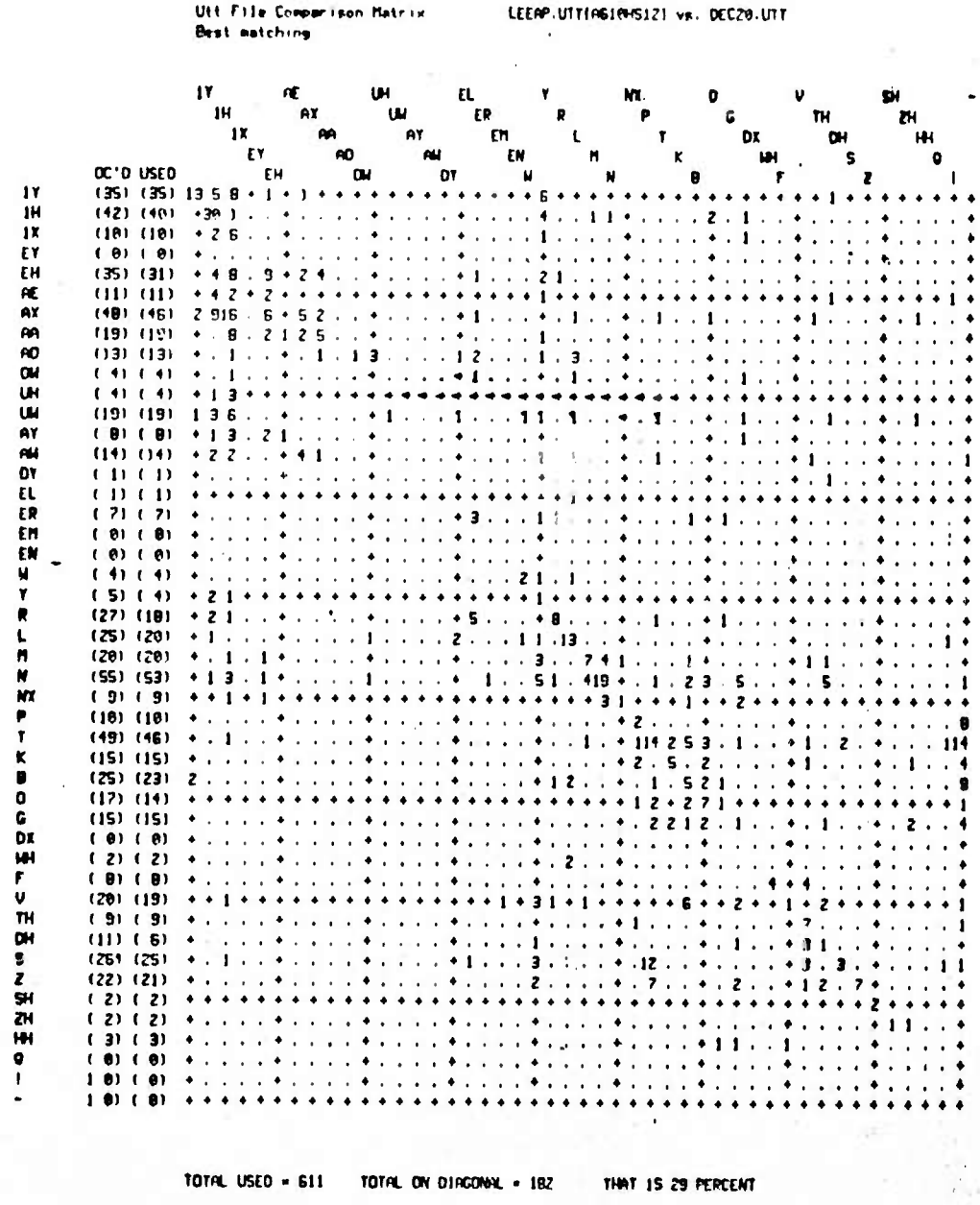
CL2:

```
VOWELS=HIGH,MID,LOW,FRNT,CNT,BAK,DIPH
RES=NAS,GL,LIQ
ST=VOST,VLST
FLAP=FLAP
FR=VOFR,VLFR
GRB=GRB
SIL=SIL
```

CL3:

```
HIGH=IY,IH,IX,UW,UH
MID=EY,EH,OW,AX,ER
LOW=AE,AA,AO
DIPH=AY,AW,OY
LIQ=L,R,EL
GL=W,Y
NAS=M,N,NX,EM,EN
VOST=B,D,G
VLST=P,T,K,Q
FLAP=DX
VOFR=DH,V,WH,Z,ZH
VLFR=TH,F,S,SH,HH
GRB=!,&
SIL=-
```

Much of the work done on the phonetic component up until this time has been in the realm of representation of knowledge and control. The feature space and pattern matching facilities have been designed in such a way that new rules, features, and labels can be handled easily. Complete output facilities are also available to aid in the analysis and formulation of rules. Future work in the phonetic component will be concentrated on improvement in the current rules and the addition of new rules. It is expected that the evaluation statistics will rapidly show improved performance as new rules are put into the system.



Utt File Comparison Matrix LEEAP.UTTIA61#HS121 vs. D.C20.UTT
Best matching

CL1

	OC'D	USED
HIGH	(89)	(88)
MID	(94)	(88)
LOW	(43)	(43)
FPNT	(C3)	(87)
CNT	(65)	(63)
BNK	(59)	(59)
DIPH	(23)	(23)
L1Q	(53)	(39)
GL	(9)	(0)
NAS	(84)	(62)
VOST	(57)	(52)
VLST	(74)	(71)
FLAP	(0)	(0)
VOFR	(57)	(50)
VLFR	(40)	(47)
GPB	(0)	(0)
SIL	(0)	(0)

TOTAL USED = 850 TOTAL ON DIAGONAL = 262 THAT IS 30 PERCENT

Utt File Comparison Matrix LEEAP.UTTIA61#HS121 vs. DEC20.UTT
Best matching

CL2

	OC'D	USED
VOWELS	(**)	(**)
RES	(E6)	(C9)
ST	(D1)	(C3)
FLAP	(0)	(0)
FR	(A5)	(97)
GPB	(0)	(0)
SIL	(0)	(0)

TOTAL USED = 850 TOTAL ON DIAGONAL = 584 THAT IS 68 PERCENT

Utt File Comparison Matrix LEEAP.UTTIA61#HS121 vs. DEC20.UTT
Best matching

CL3

	OC'D	USED
HIGH	(B0)	(88)
MID	(94)	(88)
LOW	(43)	(43)
DIPH	(23)	(23)
L1Q	(53)	(39)
GL	(9)	(8)
NAS	(84)	(82)
VOST	(57)	(52)
VLST	(74)	(71)
FLAP	(0)	(0)
VOFR	(57)	(50)
VLFP	(48)	(47)
GPB	(0)	(0)
SIL	(0)	(0)

TOTAL USED = 611 TOTAL ON DIAGONAL = 267 THAT IS 43 PERCENT

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
(18) 1. REPORT NUMBER AFOSR - TR - 76 - 0884	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) WORKING PAPERS IN SPEECH RECOGNITION. IV. The HEARSAY II System.		5. TYPE OF REPORT & PERIOD COVERED (9) Interim rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) CMU Computer Science Speech Group		8. CONTRACT OR GRANT NUMBER(s) (15) F44620-73-C-0074, ARPA Order-2466
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Dept. Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS AO 2466 61101D (12) 175 p.
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Project Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE (11) Feb 1976
		13. NUMBER OF PAGES 173
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Air Force Office of Scientific Research (NM) Bolling AFB, DC 20332		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This volume contains several papers describing the current status of the Hearsay II system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 403 181

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PRECEDING PAGE BLANK-NOT FILMED